LAB ASSIGNMENT NO – 10

Aim: Write a program in Node JS to

a. Create a file

b. Read the data from file

c. Write the data to a file

d. Rename a file

e. Append data to a file

f. Delete a file

Theory:

Node.js fs (File System) module provides an API for interacting with the file system. One of the features it offers is file streams, which allow you to read from or write to files in a more efficient and scalable manner, especially when dealing with large files or data.

1. Creating a File:

To create a file, you can use the fs.createWriteStream() method and simply open the file for writing. If the file does not exist, it will be created.

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

const filePath = 'example.txt';

const writeStream = fs.createWriteStream(filePath);

writeStream.end(() => {

  console.log('File created successfully.');

});

writeStream.on('error', (err) => {

  console.error('Error creating the file:', err);

});
```

2. Reading from a File:

To read data from a file, you can use the fs.createReadStream() method and listen for the data event to receive chunks of data.\

```
const fs = require('fs');
const filePath = 'example.txt';
const readStream = fs.createReadStream(filePath, 'utf8');
readStream.on('data', (chunk) => {
  console.log('Received chunk of data:', chunk);
});
readStream.on('end', () => {
  console.log('Finished reading the file.');
});
readStream.on('error', (err) => {
  console.error('Error reading the file:', err);
});
```

3. Writing to a File:

To write data to a file, you can use the write method on a writable stream. Data can be written in chunks.

```
const fs = require('fs');
const filePath = 'example.txt';
const writeStream = fs.createWriteStream(filePath, 'utf8');
writeStream.write('Writing the first chunk of data.\n');
writeStream.write('Writing another chunk of data.\n');
writeStream.end(() => {
  console.log('Finished writing to the file.');
});
writeStream.on('error', (err) => {
  console.error('Error writing to the file:', err);
});
```

4. Renaming a File:

To rename a file, you can use the fs.rename() method.

```
const fs = require('fs')
const oldFilePath = 'example.txt';
const newFilePath = 'new_example.txt';
fs.rename(oldFilePath, newFilePath, (err) => {
  if (err) throw err;
  console.log('File renamed successfully.');
});
```

5. Appending Data to a File:

To append data to a file, you can use the fs.appendFile() method.

```
const fs = require('fs');
const filePath = 'example.txt';
const dataToAppend = 'This data will be appended.';
fs.appendFile(filePath, dataToAppend, (err) => {
  if (err) throw err;
  console.log('Data appended to the file.');
});
```

6. Deleting a File:

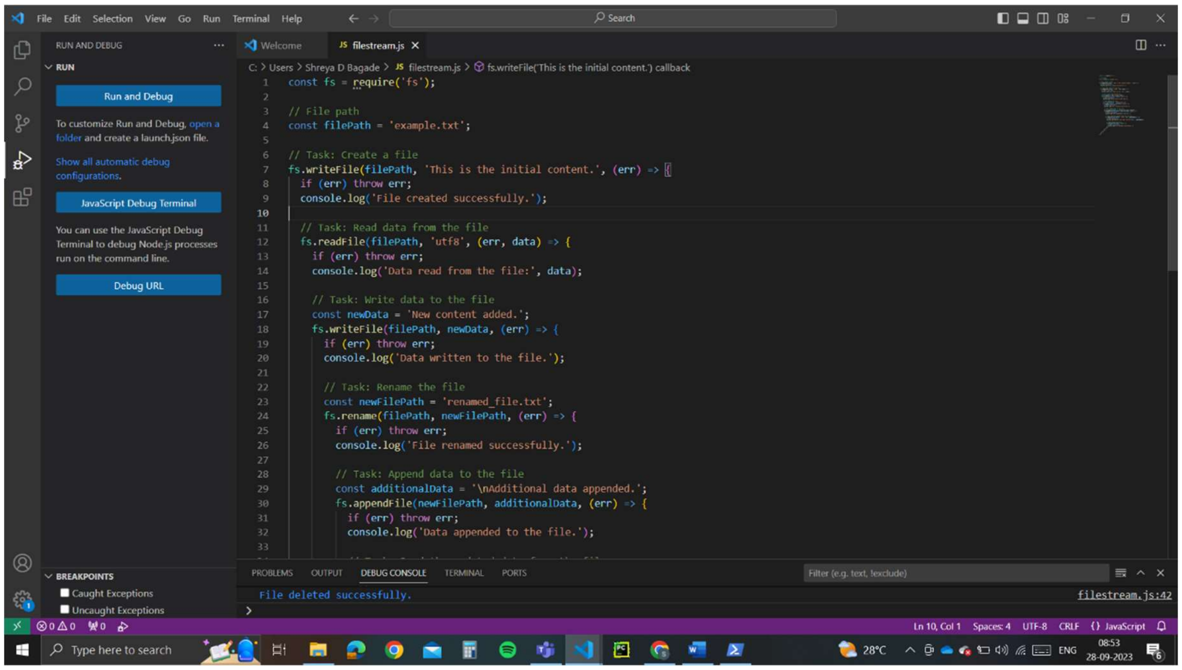
To delete a file, you can use the fs.unlink() method.

```
const fs = require('fs');
const filePath = 'example.txt';
fs.unlink(filePath, (err) => {
  if (err) throw err;
  console.log('File deleted successfully.');
});
```

Conclusion: The Node.js filestream experiment showcased the power of asynchronous I/O operations for file handling. Leveraging streams for reading, writing, appending, and manipulating files provides efficient handling of data, especially for large files. The experiment demonstrated that file streams are crucial tools for modern web applications, offering scalability and improved performance.

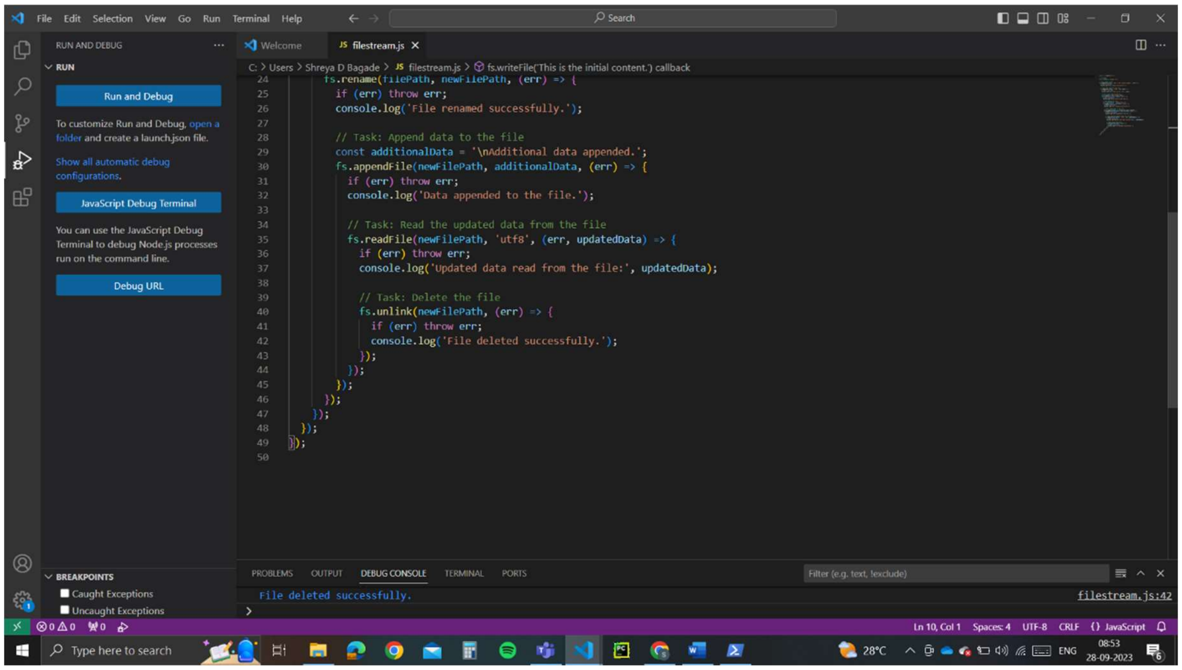
Shreya Bagade/ SEM V/ TE.IT/ T11/ Roll No-09/ Assignment no-10

<u>Code</u>:



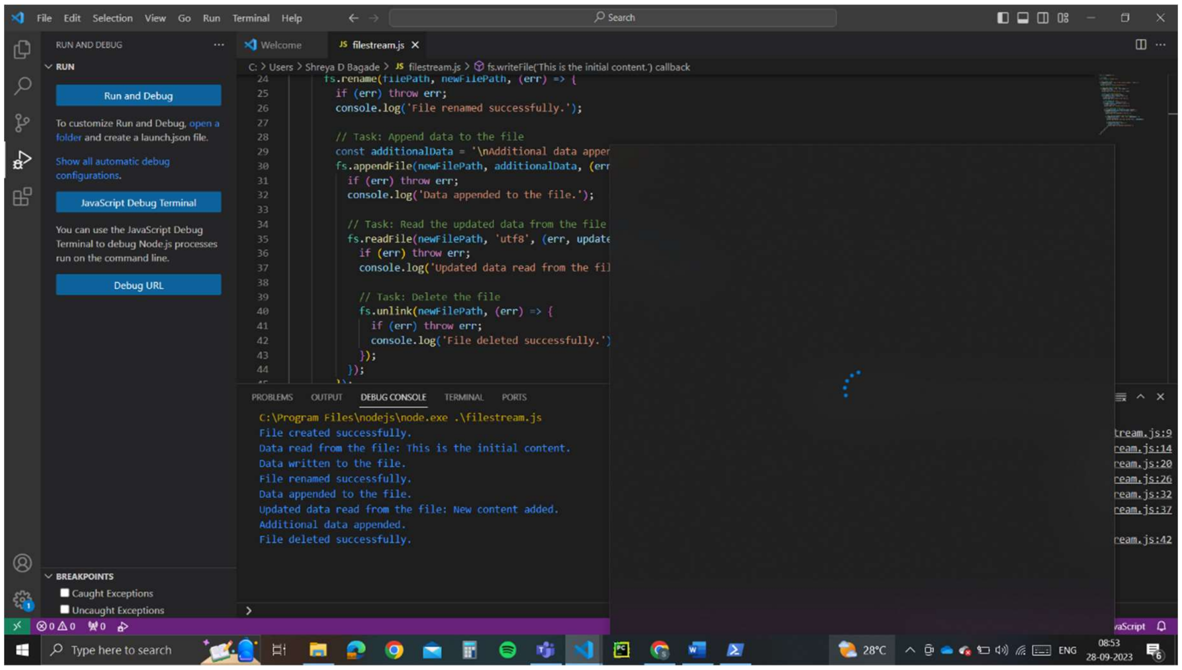
```
const fs = require('fs');

// File path
const filePath = 'example.txt';

// Task: Create a file
fs.writeFile(filePath, 'This is the initial content.', (err) => {
  if (err) throw err;
  console.log('File created successfully.');

  // Task: Read data from the file
  fs.readFile(filePath, 'utf8', (err, data) => {
    if (err) throw err;
    console.log('Data read from the file:', data);

    // Task: Write data to the file
    const newData = 'New content added.';
    fs.writeFile(filePath, newData, (err) => {
      if (err) throw err;
      console.log('Data written to the file.');

      // Task: Rename the file
      const newFilePath = 'renamed_file.txt';
      fs.rename(filePath, newFilePath, (err) => {
        if (err) throw err;
        console.log('File renamed successfully.');

        // Task: Append data to the file
        const additionalData = '\nAdditional data appended.';
        fs.appendFile(newFilePath, additionalData, (err) => {
          if (err) throw err;
          console.log('Data appended to the file.');

```

```
File deleted successfully.                                    filestream.js:42
```



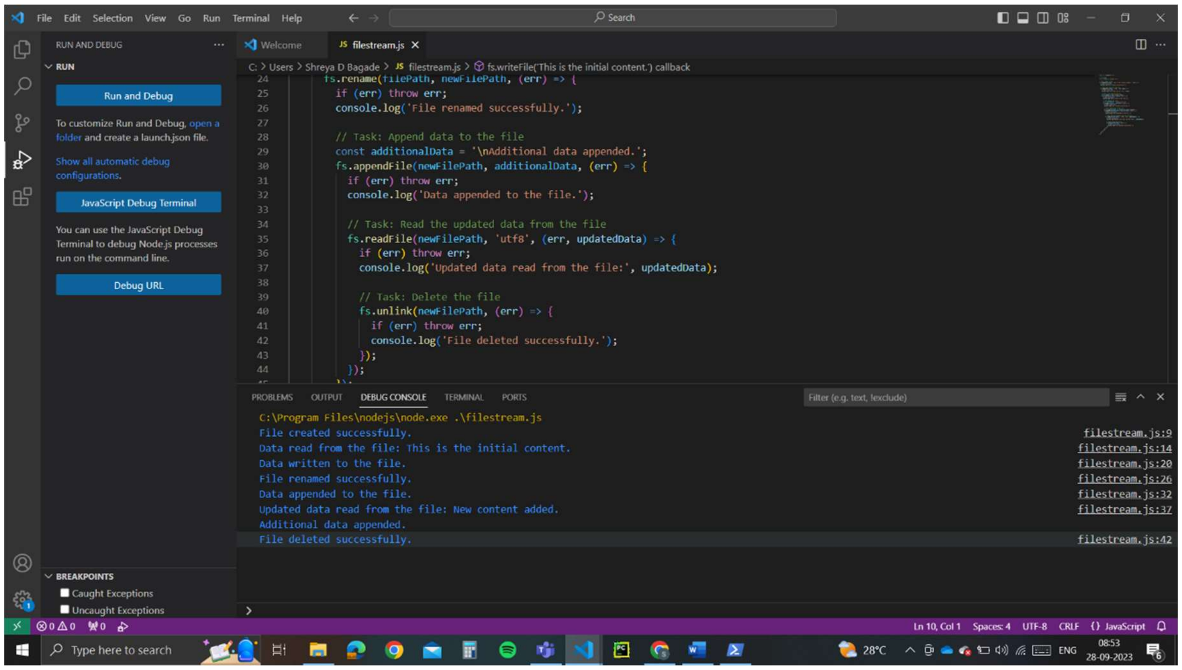
```
        fs.rename(filePath, newFilePath, (err) => {
        if (err) throw err;
        console.log('File renamed successfully.');

        // Task: Append data to the file
        const additionalData = '\nAdditional data appended.';
        fs.appendFile(newFilePath, additionalData, (err) => {
          if (err) throw err;
          console.log('Data appended to the file.');

          // Task: Read the updated data from the file
          fs.readFile(newFilePath, 'utf8', (err, updatedData) => {
            if (err) throw err;
            console.log('Updated data read from the file:', updatedData);

            // Task: Delete the file
            fs.unlink(newFilePath, (err) => {
              if (err) throw err;
              console.log('File deleted successfully.');
            });
          });
        });
      });
    });
  });
});

```

```
File deleted successfully.                                    filestream.js:42
```


RUN AND DEBUG
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
JavaScript Debug Terminal
You can use the JavaScript Debug Terminal to debug Node.js processes run on the command line.
Debug URL
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
Welcome
filestream.js
C: > Users > Shreya D Bagade > filestream.js > fs.writeFile('This is the initial content.') callback
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Type here to search
28°C ENG 08:53 28-09-2023

```
fs.rename(filePath, newFilePath, (err) => {
  if (err) throw err;
  console.log('File renamed successfully.');

  // Task: Append data to the file
  const additionalData = '\nAdditional data apper
  fs.appendFile(newFilePath, additionalData, (err
    if (err) throw err;
    console.log('Data appended to the file.');

    // Task: Read the updated data from the file
    fs.readFile(newFilePath, 'utf8', (err, update
      if (err) throw err;
      console.log('Updated data read from the fil

      // Task: Delete the file
      fs.unlink(newFilePath, (err) => {
        if (err) throw err;
        console.log('File deleted successfully.')
      });
    });
```

```
C:\Program Files\nodejs\node.exe .\filestream.js
File created successfully.                                  tream.js:9
Data read from the file: This is the initial content.       ream.js:14
Data written to the file.                                   ream.js:20
File renamed successfully.                                  ream.js:26
Data appended to the file.                                  ream.js:32
Updated data read from the file: New content added.         ream.js:37
Additional data appended.
File deleted successfully.                                  ream.js:42
```


RUN AND DEBUG
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
JavaScript Debug Terminal
You can use the JavaScript Debug Terminal to debug Node.js processes run on the command line.
Debug URL
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
Welcome
filestream.js
C: > Users > Shreya D Bagade > filestream.js > fs.writeFile('This is the initial content.') callback
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Filter (e.g. text, !exclude)
Ln 10, Col 1 Spaces: 4 UTF-8 CRLF JavaScript
Type here to search
28°C ENG 08:53 28-09-2023

```
fs.rename(filePath, newFilePath, (err) => {
  if (err) throw err;
  console.log('File renamed successfully.');

  // Task: Append data to the file
  const additionalData = '\nAdditional data appended.';
  fs.appendFile(newFilePath, additionalData, (err) => {
    if (err) throw err;
    console.log('Data appended to the file.');

    // Task: Read the updated data from the file
    fs.readFile(newFilePath, 'utf8', (err, updatedData) => {
      if (err) throw err;
      console.log('Updated data read from the file:', updatedData);

      // Task: Delete the file
      fs.unlink(newFilePath, (err) => {
        if (err) throw err;
        console.log('File deleted successfully.');
      });
    });
```

```
C:\Program Files\nodejs\node.exe .\filestream.js
File created successfully.                                  filestream.js:9
Data read from the file: This is the initial content.       filestream.js:14
Data written to the file.                                   filestream.js:20
File renamed successfully.                                  filestream.js:26
Data appended to the file.                                  filestream.js:32
Updated data read from the file: New content added.         filestream.js:37
Additional data appended.
File deleted successfully.                                  filestream.js:42
```

<u>Lab Outcome :</u> LO-6: Construct back end applications using Node.js/Express.