SE3040 -Application Frameworks

Lab Exercise 4

Software Engineering - Year - 3 **Semester 1 - 2022**

Lab Session: NodeJS

Objective: Teach main features of NodeJS runtime + JavaScript library

1. Hello world.
    a. Check the node version of the machine by issuing the following command. `node -version`

    b. Create a file named app.js.

    c. Use default console log command to print string to the console.
    ```
    console.log('Hello World');
    ```

    d. Type the following command in the command prompt (Command should be opened in the working directory).
    node app.js

2. Use OS system library.
    a. Import the OS system module to your file.
    ```
    const os = require('os');
    ```

    b. Obtain System architecture, platform and number of CPUs from the OS module and print them to the console.
    ```
    console.log('Architecture ' + os.arch());
    console.log('CPUs ' + os.cpus().length);
    console.log('OS ' + os.platform());
    ```

    c. Run application and check the output.

3. Read a file.
    a. Create a file named test.txt and add the following content.
    `NodeJS is awesome.`

    b. Import the fs system module to read the file.
    ```
    const fs = require('fs');
    ```

c. Use the system variable __dirname to set the file location.

```
const fileName = __dirname + '/test.txt';
```

d. Read the file using readFile asynchronous method and print the content of the file to console.

```
fs.readFile(fileName, (err, data) => {
    if (err) {
        console.error(err);
    }

    console.log(data.toString());
});
```

e. Try printing the value of data without toString method.

f. Use the readFileSync method to read the file synchronously.

```
const data = fs.readFileSync(fileName);
console.log(data.toString());
```

4. Use streams to copy content of a file.
   a. Add two variables containing path to the source and destination files.

```
const fileName = __dirname + '/test.txt';
const outFileName = __dirname + '/test-copy.txt';
```

   b. Create read stream and write stream from the source file and destination file respectively.

```
const readStream =
fs.createReadStream(fileName); const writeStream =
fs.createWriteStream(outFileName);
```

   c. Pipe the read stream to write stream.

```
readStream.pipe(writeStream);
```

   d. Verify the file named test-copy.txt is being created with the same content as the test.txt.

   e. Optionally listen to the data event of the read stream and print the output.

```
readStream.on('data', data => {
    console.log(data.toString());
});
```

5. Http Server
    a. Import the http module from the core libraries.

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

    b. Create a httpServer that listens to port 3000 and return HTML with Hello World text in h1 header upon GET request.

```
http.createServer((req, res) => {
res.setHeader('Content-Type', 'text/html');
        res.write('<h1>Hello World</h1>');
        res.end();
}).listen(3000);
```

    c. Check the output of the code by running the code and accessing http://localhost:3000 in browser.

    d. Optionally add a POST request that accepts form field name and return HTML with Hello {name}.

```
const http = require('http');
http.createServer((req, res) => {

    res.setHeader('Content-Type', 'text/html');

    switch (req.method) {        case 'GET':
    res.write('<h1>Hello World</h1>');
    res.end();            break;        case
    'POST':
            req.on('data', data => {
    res.write('<h1>Hello ' + data + '</h1>');
    res.end();
            });
    break;
    }

    }).listen(3000, (err) => {
    console.log('Server is listening to port 3000')
    });
```