WEEK-4 (Nodejs)

Task-1

URL Parsing and Manipulation

Aim1: Write a program that accepts a URL as user input and uses the url module to parse it. Display the protocol, host, path, and query parameters separately.

Theoretical Background:

URL stands for Uniform Resource Locator. A URL is nothing more than the address of a given unique resource on the Web. In theory, each valid URL points to a unique resource. Such resources can be an HTML page, a CSS document, an image, etc. In practice, there are some exceptions, the most common being a URL pointing to a resource that no longer exists or that has moved.

Source Code:

```
const { hostname } = require("os");
const readline = require("readline");
const url = require('url');
const a = readline.createInterface({
    input: process.stdin,
    output: process.stdout     });
a.question("Enter the you want:", (inputUrl)=>{
    const parsedUrl = url.parse(inputUrl, true);
    console.log('Protocol:', parsedUrl.protocol);
    console.log('Host:', parsedUrl.host);
    console.log('Path:', parsedUrl.pathname);
    console.log('Query Parameters:', parsedUrl.query);
    a.close();});
```

Output:

```
PS C:\Users\Dell\OneDrive\Desktop\fswd> node p1.js
Enter the you want:https://www.youtube.com/watch?v=x86ivk2Sgo0
Protocol: https:
Host: www.youtube.com
Path: /watch
Query Parameters: [Object: null prototype] { v: 'x86ivk2Sgo0' }
```

Aim2: Implement a function that takes a base URL and a relative path as input, and uses the url module to resolve and display the absolute URL.

Theoretical Background:

URL stands for Uniform Resource Locator. A URL is nothing more than the address of a given unique resource on the Web. In theory, each valid URL points to a unique resource. Such resources can be an HTML page, a CSS document, an image, etc.

Source Code:

```
const url = require('url');
function resolveURL(baseUrl, relativePath) {
    const resolvedUrl = new URL(relativePath, baseUrl);
    console.log('Absolute URL:', resolvedUrl.href);
}
// Usage example:
resolveURL('https://classroom.google.com/c/NjE1MTU3NDMyMTE5/a/NjE2MjU5MzE0
ODEw/details', 'about');
```

Output:

```
PS C:\Users\Dell\OneDrive\Desktop\fswd> node p1.js
Absolute URL: https://classroom.google.com/c/NjE1MTU3NDMyMTE5/a/NjE2MjU5MzE0ODEw/about
```

Task-2

Query String Operation

Aim1: Write a Node.js program that takes a URL with a query string as input and extracts the key-value pairs from the query string using the querystring module. The program should display the extracted key-value pairs as output.

Theoretical Background:

A query string is a part of a URL that follows a question mark (?) and is used to pass data to a web page or application. It is typically composed of key-value pairs that are separated by an ampersand (&). The key represents the parameter and the value represents the data that is being passed through the parameter.

Source Code:

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

const url = require('url');

let q = url.parse('http://www.company.com?user=Mahek&year=2003');
let qstring = querystring.parse(q.query);

console.log(qstring);
```

Output:

```
PS C:\Users\DELL\OneDrive\Documents\Nodejs> node t1.js
[Object: null prototype] { user: 'student', year: '2003' }
PS C:\Users\DELL\OneDrive\Documents\Nodejs>
```

Task-3 Path Operations

Aim1: Create a program that accepts two file paths as input and uses the path module to determine if they refer to the same file.

Theoretical Background:

The path module in node js is used for managing and altering file paths. It's a part of the core modules with methods that help you deal with directory and file path names on the local machine's file system. Every system has a different environment and a specific operating system to manage it.

Source Code:

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

const path = require('path');

function areSameFiles(filePath1, filePath2) {

  const absolutePath1 = path.resolve(filePath1);

  const absolutePath2 = path.resolve(filePath2);

  return absolutePath1 === absolutePath2; }
// Example usage
const file1 = 'Downloads/fswd/test2.js';

const file2 = 'Downloads/fswd/test3.js';

if (areSameFiles(file1, file2)) {
```

```
  console.log(`refer to the same file.`);

} else {

  console.log(`do not refer to the same file.`);   }
```

Output:

```
PS C:\Users\Dell\OneDrive\Desktop\fswd> node p1.js
do not refer to the same file.
```

Aim2: Implement a function that accepts a file path as input and uses the path module to extract the file extension. Display the extracted extension to the user.

Theoretical Background:

Path module has a basename method to get the file name part and usage of the same function in Windows and POSIX(I.e. Unix) differs. Node. js path can be used to get consistent results irrespective of Operating Systems with the help of specific implementation methods.

Source Code:

```
const path = require('path');
function extractFileExtension(filePath) {
  const extension = path.extname(filePath);
  return extension;}
// Example usage
const filePath = '/Downloads/fswd/task2.html'; const
fileExtension = extractFileExtension(filePath);
console.log(`The file extension is: ${fileExtension}`);
```

Output:

```
PS C:\Users\Dell\OneDrive\Desktop\fswd> node p1.js
The file extension is: .html
```

Task-4

File Paths and Operations

Aim1: Implement a program that accepts a file path as input and uses the path module to extract the directory name and base name. Display the extracted values separately.

Theoretical Background:

To handle file operations like creating, reading, deleting, etc., Node.js provides an inbuilt module called FS (File System). Node.js gives the functionality of file I/O by providing wrappers around the standard POSIX functions. All file system operations can have synchronous and asynchronous forms depending upon user requirements.

Source Code:

```
const path = require('path');
function extractDirectoryAndBaseName(filePath) {
  const directoryName = path.dirname(filePath);
  const baseName = path.basename(filePath);
  return { directoryName, baseName };}
// Example usage
const filePath = '/Downloads/fswd/test7.js';
const { directoryName, baseName } = extractDirectoryAndBaseName(filePath);
console.log(`Directory Name: ${directoryName}`);
console.log(`Base Name: ${baseName}`);
```

Output:

```
PS C:\Users\Dell\OneDrive\Desktop\fswd> node p1.js
Directory Name: /Downloads/fswd
Base Name: test7.js
```

Aim2: Write a function that uses the fs module to check if a given file path exists. Display a success message if the file exists, or an error message if it doesn't.

Theoretical Background:

Every file in the system has a path. On Linux and macOS, a path might look like: /users/joe/file.txt while Windows computers are different, and have a structure such as:
C:\users\joe\file.txt

Source Code:

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

function checkFileExists(filePath) {

  fs.access(filePath, fs.constants.F_OK, (err) => {
```

```
    if (err) {
      console.error(`Error: The file "${filePath}" does not exist.`);
    } else {
      console.log(`Success: The file "${filePath}" exists.`); } });    }
// Example usage
const filePath = 'C:/Users/Meghavi Lad/Downloads/fswd/task2.html';
checkFileExists(filePath);
```

Output:

```
PS C:\Users\Dell\OneDrive\Desktop\fswd> node p1.js
Success: The file "C:/Users/Dell/OneDrive/Desktop/fswd/week5-1.js" exists.
```

Learning Outcome:

Here, I am learning about how to url and how to manipulate urls,query string operation,path operation and also try to study about how to manage file paths in node js.How to get input from the user ,how to get information about host,path and protocol from the particular url.