WEEK 4
Task-1

Aim: URL Parsing and Manipulation:

● 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.

● 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.

Description:

The URL parsing functions focus on splitting a URL string into its components, or on combining URL components into a URL string. Parse a URL into six components, returning a 6-item named tuple. This corresponds to the general structure of a URL: scheme://netloc/path;parameters?query#fragment . Each tuple item is a string, possibly empty.

Source Code:

```
const readline = require('readline');
const url = require('url');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.question('Enter a URL: ', (inputURL) => {
  const parsedURL = url.parse(inputURL, true);

  console.log('Protocol:', parsedURL.protocol);
  console.log('Host:', parsedURL.host);
  console.log('Path:', parsedURL.pathname);

  if (Object.keys(parsedURL.query).length > 0) {
    console.log('Query Parameters:');
    for (const key in parsedURL.query) {
      console.log(`${key}: ${parsedURL.query[key]}`);
    }
  } else {
    console.log('No Query Parameters');
  }

  rl.close();
});
```

Output:

```
PS C:\Users\DELL\Desktop\sem 5\fswd\week4> node task1.js
Enter a URL: https://mail.google.com/mail/u/0/#inbox
Protocol: https:
Host: mail.google.com
Path: /mail/u/0/
No Query Parameters
```

Source Code:

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

function resolveURL(baseUrl, relativePath) {
  const absoluteUrl = url.resolve(baseUrl, relativePath);
  console.log('Absolute URL:', absoluteUrl);
}

resolveURL('https://example.com/base/path', '/relative/path');
```

Output:

```
PS C:\Users\DELL\Desktop\sem 5\fswd\week4> node task2.js
Absolute URL: https://example.com/relative/path
PS C:\Users\DELL\Desktop\sem 5\fswd\week4>
```

Task-2

Aim: Query String Operation:

● 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.

Description:

A query string is the portion of a URL where data is passed to a web application and/or back-end database. The reason we need query strings is that the HTTP protocol is stateless by design. For a website to be anything more than a brochure, you need to maintain state **.**

Source Code:

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

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// Function to extract and display key-value pairs from a query string
function extractQueryParams(inputURL) {
  const parsedURL = url.parse(inputURL);
  const queryParams = querystring.parse(parsedURL.query);

  console.log('Query Parameters:');
  for (const key in queryParams) {
    console.log(key + ':', queryParams[key]);
  }
}

// Get user input
rl.question('Enter a URL with a query string: ', (userURL) => {
  extractQueryParams(userURL);
  rl.close();
});
```

Output:

```
PS C:\Users\DELL\Desktop\sem 5\fswd\week4> node task2.js
Enter a URL with a query string: https://in.search.yahoo.com/search?fr=mcafee&type=E210IN105G0&p=courseera
Query Parameters:
fr: mcafee
type: E210IN105G0
p: courseera
```

Task-3

Aim: Path Operations:

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

● 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.

Description:

The path. join() method joins the specified path segments into one path. You can specify as many path segments as you like. The specified path segments must be strings, separated by comma.

Source Code:

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

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// Function to check if two file paths refer to the same file
function checkSameFile(path1, path2) {
  const absolutePath1 = path.resolve(path1);
  const absolutePath2 = path.resolve(path2);

  if (fs.existsSync(absolutePath1) && fs.existsSync(absolutePath2)) {
    const isSameFile = fs.statSync(absolutePath1).ino === fs.statSync(absolutePath2).ino;
    console.log('Are they the same file?', isSameFile);
  } else {
    console.log('One or both file paths are invalid.');
  }
}

// Function to extract and display the file extension from a file path
function extractFileExtension(filePath) {
  const fileExtension = path.extname(filePath);
  console.log('File Extension:', fileExtension);
}

// Get user input
rl.question('Enter the first file path: ', (path1) => {
  rl.question('Enter the second file path: ', (path2) => {
    // Check if the file paths refer to the same file
    checkSameFile(path1, path2);

    // Example usage of extractFileExtension function
    const exampleFilePath = '/path/to/example/file.txt';
    extractFileExtension(exampleFilePath);
```

```
    rl.close();
  });
});
```

Output:

```
PS C:\Users\DELL\Desktop\sem 5\fswd\week4> node task3.js
Enter the first file path: C:\Users\DELL\Desktop\sem 5\fswd
Enter the second file path: C:\Users\DELL\Desktop\sem 5\fswd
Are they the same file? true
File Extension: .txt
PS C:\Users\DELL\Desktop\sem 5\fswd\week4>
```

Task-4

Aim: File Paths and Operations:

● 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.

● 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.

Description:

path () Function. The app. path () function returns the canonical path of the app as a string.We can get the path of the present script in node. js by using __dirname and __filename module scope variables. __drimane: It returns the directory name of the current module in which the current script is located. __filename: It returns the file name of the current module.

Source Code:

```
const path = require('path');
const fs = require('fs');
// Function to extract directory name and base name
function extractFileInfo(filePath) {
const dirName = path.dirname(filePath);
const baseName = path.basename(filePath);
return {dirName, baseName};
}
// Function to check if the file exists
function checkFileExists(filePath) {
return new Promise((resolve) => {
fs.stat(filePath, (err, stats) => {
if (err) {
    resolve(false); // File does not exist
} else {
resolve(stats.isFile());
} });
});
}
const filePathToProcess = 'C:\Users\DELL\Desktop\sem 5\fswd\week4\task4.js';
// Extract directory name and base name
const fileInfo = extractFileInfo(filePathToProcess);
console.log('Directory Name:', fileInfo.dirName);
console.log('Base Name:', fileInfo.baseName);
// Check if the file exists
checkFileExists(filePathToProcess)
.then((fileExists) => {
if (fileExists) {
console.log('File does not exist!');
} else {
console.error('File exist.');
}
})
.catch((err) => {
console.error('Error:', err.message);
});
```

Output:

```
PS C:\Users\DELL\Desktop\sem 5\fswd\week4> node task4.js
Directory Name: C:
Base Name: UsersDELLDesktopsem 5
swdweek4        ask4.js
File exist.
PS C:\Users\DELL\Desktop\sem 5\fswd\week4>
```

Course Outcome:

By performing this practical I learnt How to get input from the user ,how to get information about host,path and protocol from the particular url and also try to study about how to manage file paths in node js and different types of modules and paths and Query String Operation.

CO2: Apply a deep knowledge of MVC(ModelViewController) architecture, making the development process easier and faster using open-source technologies.