**Full Stack Development (MCNE262)**

**REPL and Modules Assignment**

**RAJBALA**

**1MS24SCN13**

**Assignment 1: Named Exports with Aliases**

Objective: Practice using named exports and aliases during import.

Question:

Create a module mathOperations.js that exports the following functions using named exports:

add(a, b)

subtract(a, b)

In main.js, import these functions using aliases (e.g., add as sum, subtract as diff) and log the results of sample calculations.

**mathOperation.js**

export function add(a, b) {

return a + b;

}

export function subtract(a, b) {

return a - b;

}

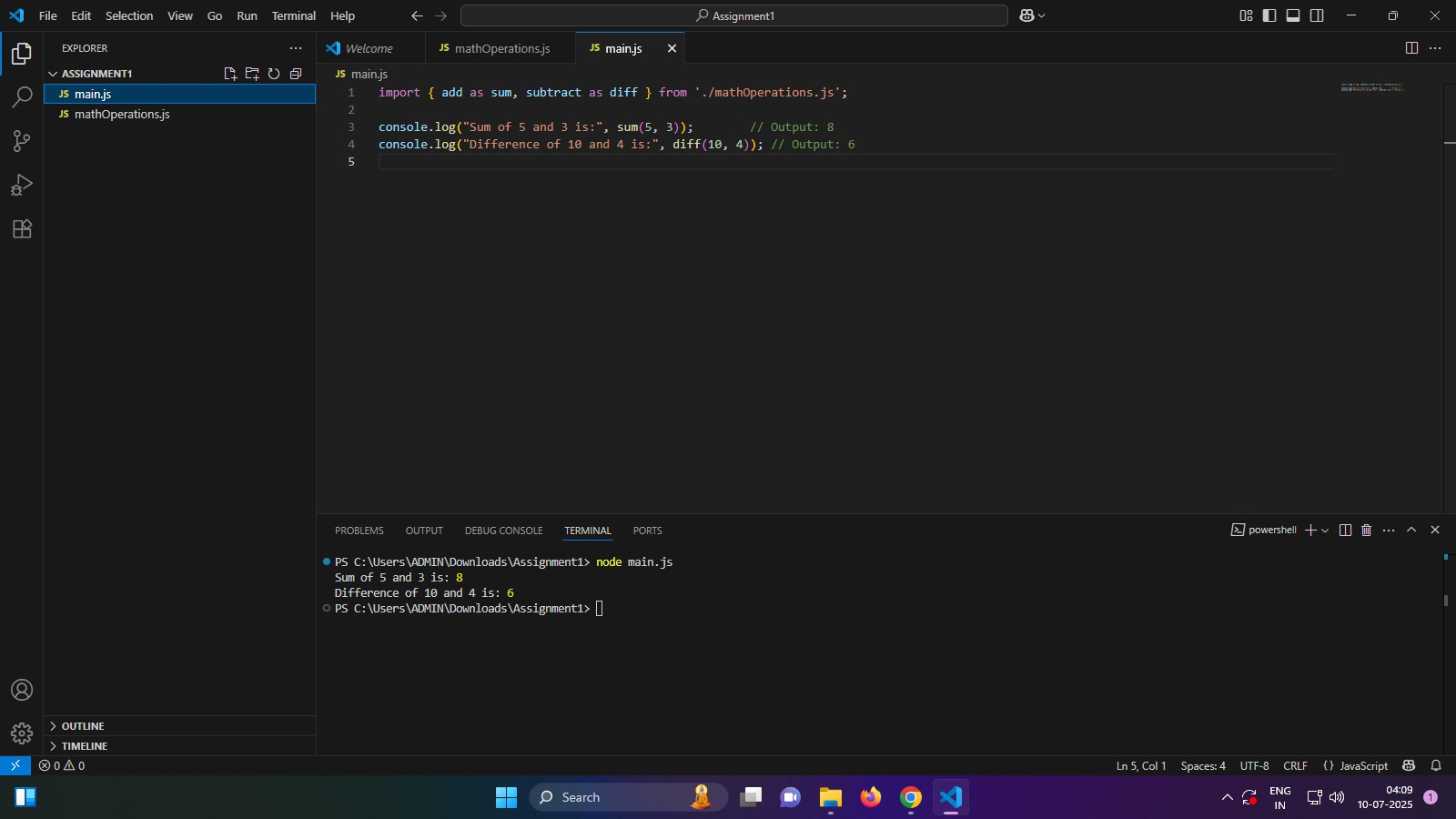
**main.js**

import { add as sum, subtract as diff } from './mathOperations.js';

console.log("Sum of 5 and 3 is:", sum(5, 3));

console.log("Difference of 10 and 4 is:", diff(10, 4));

**Output:**

****

**Assignment 2: Dynamic Imports**

Objective: Understand how to load modules at runtime using import().

Question:

Create a module stringUtils.js that exports a function capitalize(word).

In main.js, use dynamic import to load stringUtils.js only when the user inputs a word. Display the capitalized word using the imported function.

**stringUtils.js**

export function capitalize(word) {

if (!word) return '';

return word.charAt(0).toUpperCase() + word.slice(1).toLowerCase();

}

**main.js**

import readline from 'readline';

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.question('Enter a word: ', async (word) => {

if (word) {

try {

const stringUtils = await import('./stringUtils.js');

const capitalized = stringUtils.capitalize(word);

console.log(`Capitalized: ${capitalized}`);

} catch (err) {

console.error('Error importing module:', err);

}

} else {

console.log('No word entered.');

}

rl.close();

});

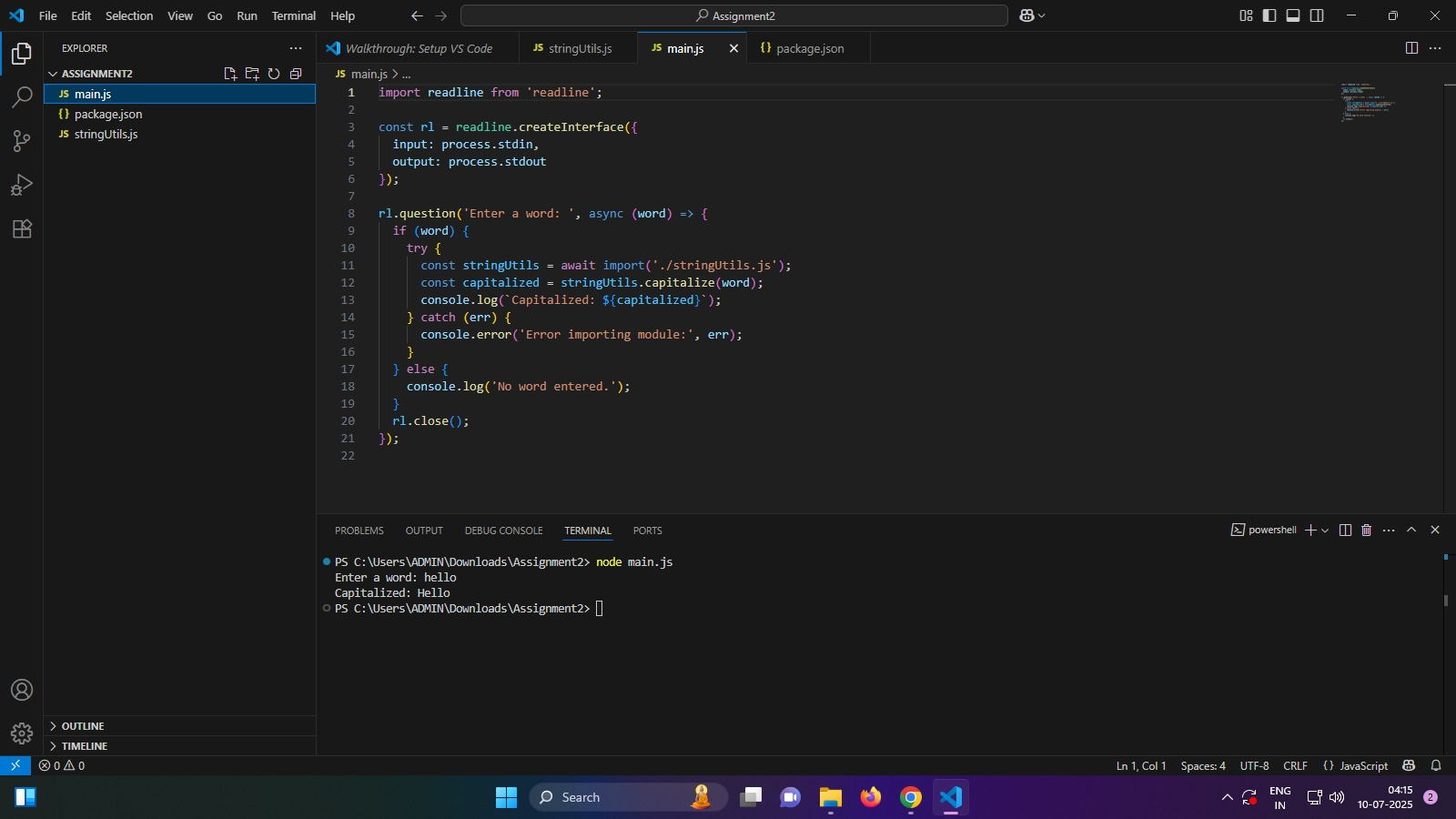
**package.json**

{

"type": "module"

}

**Output:**



**Assignment 3: Combining Default and Named Exports**

Objective: Learn how to combine default and named exports in the same module.

Question:

Create a module user.js that:

Exports a default class User with properties like name and a method getInfo().

Also exports a named function validateEmail(email).

In main.js, import the class and the function, create a User instance, and validate a sample email address using the function.

**user.js**

export default class User {

  constructor(name, email) {

    this.name = name;

    this.email = email;

  }

  getInfo() {

    return `Name: ${this.name}, Email: ${this.email}`;

  }

}

export function validateEmail(email) {

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

  return emailRegex.test(email);

}

**main.js**

import User, { validateEmail } from './user.js';

const newUser = new User('Rajbala', 'rajbalakhicher17@gmail.com');

console.log(newUser.getInfo());

const sampleEmail = 'rajbalakhicher17@gmail.com';

console.log(`Is "${sampleEmail}" a valid email?`, validateEmail(sampleEmail));

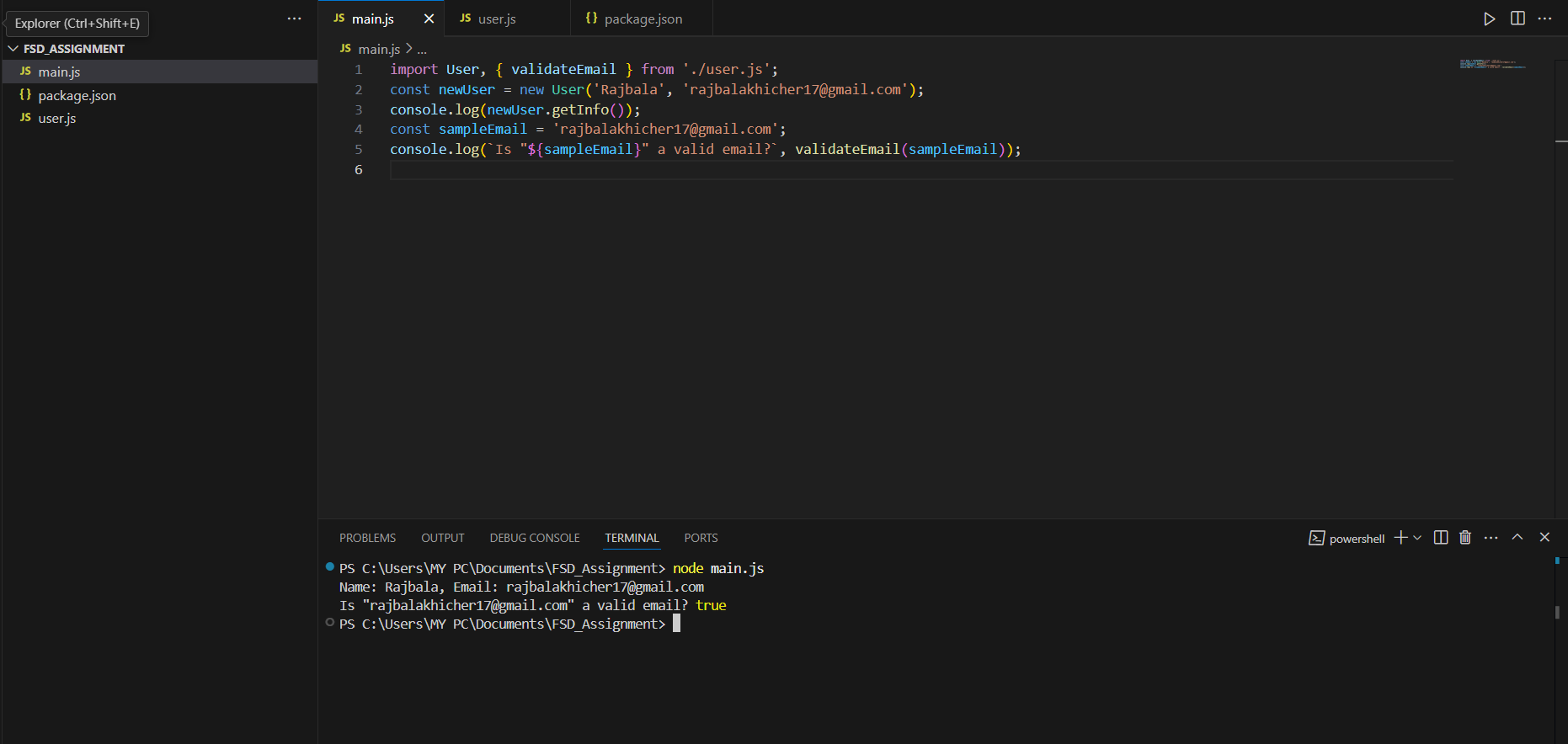
**package.json**

{

    "type": "module"

}

**Output:**

****

**REPL**

Write a JavaScript program that:

Takes a predefined array of integers.

Uses a loop to iterate through the array.

Checks each number to see if it is even.

Calculates the sum of all even numbers in the array.

Prints the final sum.

**main.js**

let numbers = [5, 8, 12, 3, 7, 10, 4];

let sum = 0;

for (let i = 0; i < numbers.length; i++) {

  if (numbers[i] % 2 === 0) {

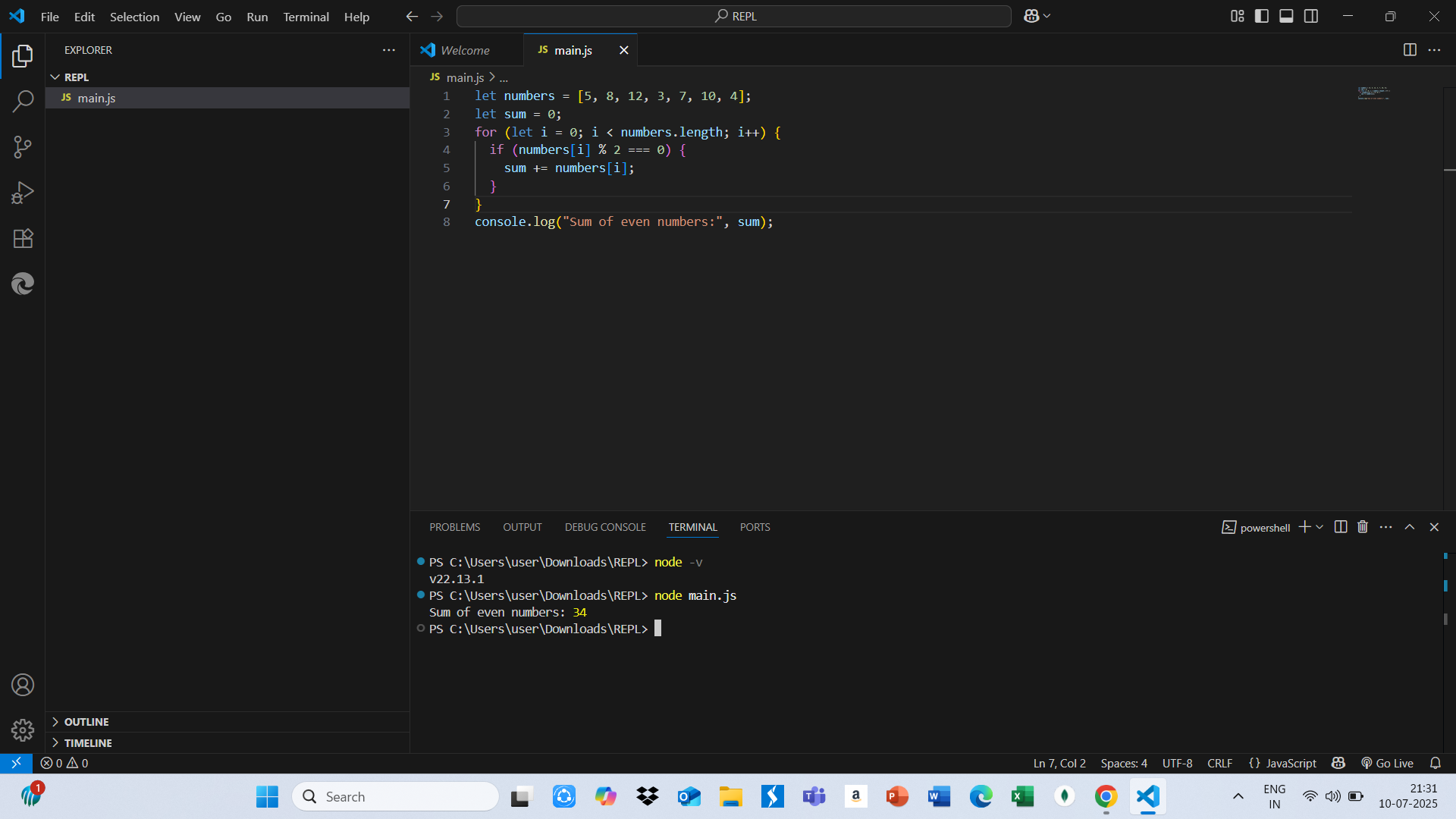
    sum += numbers[i];

  }

}

console.log("Sum of even numbers:", sum);

**Output:**



**Explanation:**

**REPL** stands **for Read-Eval-Print Loop,** and it is an interactive programming environment that takes user input, reads it, evaluates(executes) the code, prints the result, and then loops back to wait for more input. It allows you to test code snippets quickly without the need to write, save, and run full programs. In the case of JavaScript or Python REPL, when you enter a line of code (like a loop, condition, or function), the REPL runs it immediately and shows the result right away. This makes it a powerful tool for beginners learning how each line of code works, and for developers to debug or experiment with logic interactively