Portfolio for CN5006: Web and Mobile Application Development

1. Reflection on the First Week:

- - Downloaded and explored tools: Postman, MongoDB, and Node.js.
  - Understood the differences and uses of these tools.
  - Learned to compile a program using Visual Studio Code.
  - Practiced using `console.log` for debugging in code.
- I learn the differences between postman, MongoDB and Nodejs
- I learn the uses of it.
- I learn how to compile a program in a visual studio code.
- I learn the uses of console.log in code.

2. Reflection on Understanding and Writing JavaScript Code:

Working with JavaScript to solve problems was insightful. It involved understanding syntax and applying logical structures to achieve desired functionalities. Writing programs, like a division program with `console.log` feedback, deepened my understanding of JavaScript's capabilities.

- Coding of multiple, addition , subtraction with output.

JS index.js > ...

```js
const num1 =10;
const num2 = 5;
var sum1 = num1+num2
//Addition calulation
let sum = num1 +num2;

console.log("The sum of",num1 ,"and", num2, "is", sum);
//substarction calculation
const substarction = num1-num2;
console.log("The substraction  of",num1 ,"and", num2, "is", substarction);
//multiplication
const multiplication = num1*num2;
console.log("The multiplication of ",num1, "and",num2, "is", multiplication
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
The substraction  of 10 and 5 is 5
PS C:\Users\user\Desktop\index js> node index.js
The sum of 10 and 5 is 15
The substraction  of 10 and 5 is 5
The multiplication of  10 and 5 is 50
PS C:\Users\user\Desktop\index js> node index.js
The sum of 10 and 5 is 15
The substraction  of 10 and 5 is 5
The multiplication of  10 and 5 is 50
```

powershell
powershell

Week 2 MongoDB

1)Write MongoDB queries for the following using either command shell:

1. Repeat the same process to search Education for Master and Find the avg ,min, max age and avg min max Salary of the people group by Marital status.

Stage 1 $match

```
/**
 * query: The query in MQL.
 */
{
  Education:"Master",

}
```

Output after $match stage (Sample of 10 documents)

```
_id: ObjectId('67113be28c32df5a77b30fad')
First Name : "Evelyn"
Last Name : "Wells"
Gender : "Female"
Age : 24
Email : "e.wells@randatmail.com"
Education : "Master"
Salary : 2923
Marital Status : "Single"
```

```
_id: ObjectId('67113be28c
First Name : "Martin"
Last Name : "Alexander"
Gender : "Male"
Age : 26
Email : "m.alexander@randa
Education : "Master"
Salary : 2739
Marital Status : "Single"
```

Stage 2 $group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: "$Gender",
  Avg:{$avg:"$Age"},
  MinAge:{$min:"$Age"},
  MaxAge:{$max:"$Age"},
  }
```

Output after $group stage (Sample of 2 documents)

```
_id: "Male"
Avg : 26.22222222222222
MinAge : 22
MaxAge : 30
```

```
_id: "Female"
Avg : 26.692307692307693
MinAge : 22
MaxAge : 30
```

Stage 2 $group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: "$Marital Status",
  Avg:{$avg:"$Age"},
  MinAge:{$min:"$Age"},
  MaxAge:{$max:"$Age"},
  MaxSalary:{$max:"$Salary"},
  MinSalary:{$max:"$Salary"},
  AvgSalary:{$avg:"$salary"}
  }
```

Output after $group stage (Sample of 2 documents)

```
_id: "Married"
Avg : 27.625
MinAge : 22
MaxAge : 30
MaxSalary : 8430
MinSalary : 8430
AvgSalary : null
```

```
_id: "Single"
Avg : 25.857142857142858
MinAge : 22
MaxAge : 30
MaxSalary : 8722
MinSalary : 8722
AvgSalary : null
```

2)find min,max average salary of each age group of female

3.)find min,max average salary of each age group of male

Stage 3 $group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */

  {
  _id: "$Gender",
  Avg:{$avg:"$Age"},
  MinAge:{$min:"$Age"},
  MaxAge:{$max:"$Age"},
  MaxSalary:{$max:"$Salary"},
  MinSalary:{$max:"$Salary"},
  AvgSalary:{$avg:"$salary"}
  }
```

Output after $group stage (Sample of 2 documents)

```
_id: "Female"
Avg : 26.692307692307693
MinAge : 22
MaxAge : 30
MaxSalary : 8722
MinSalary : 8722
AvgSalary : null
```

```
_id: "Male"
Avg : 26.22222222222222
MinAge : 22
MaxAge : 30
MaxSalary : 8430
MinSalary : 8430
AvgSalary : null
```

find min, max average salary of each age group of female

3. find min, max average salary of each age group of male

4. Count married and unmarried females and males.

write a reflective report summarizing your lab work for today. Include screenshots of the MongoDB queries you performed.

= In this lab, I learned how to use MongoDB for various tasks like inserting, updating, and deleting documents. I faced some challenges, especially with the syntax in aggregation pipelines, such as using `$match` and `$group` stages, but by referring to MongoDB documentation and practicing with sample queries, I was able to overcome them. I also learned how to perform complex queries to calculate averages, minimum and maximum values, and group data. Overall, this lab helped me understand how to manage data effectively using MongoDB's flexible schema and powerful aggregation features, which are great for working with different types of data.

Week 3

Welcome × | JS Employeeinfo.js U ×

JS Employeeinfo.js > ...

```javascript
// Definition of the function EmployeeInfo
function EmployeeInfo(name, Salary) {
    console.log("Welcome " + name + " Your monthly Salary is " + Salary);
}

console.log("This is my first program");

var EmpName = "John";
var EmpSalary = 50000;

// Calling the function EmployeeInfo
EmployeeInfo(EmpName, EmpSalary);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER powershell

```
PS C:\Users\user\Desktop\index js> node Employeeinfo.js
This is my first program
Welcome John Your monthly Salary is 50000
PS C:\Users\user\Desktop\index js>
```

Code for creating arrow function

```
//code for creating arrow function
const EmpSkills= (skills)=> {
    console.log("Expert in "+ skills)
   }
   EmpSkills("java")
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    SPELL CHECKE

Welcome John Your monthly Salary is 50000
PS C:\Users\user\Desktop\index js> node Employeeinfo.js
This is my first program
Welcome John Your monthly Salary is 50000
Expert in java
PS C:\Users\user\Desktop\index js>
```

Index.js where I import student and person code.

```
Welcome   Employeeinfo.js U   StudentInfo.js M   person.js M   index.js M
index.js > ...
// Index.js
const student = require('./StudentInfo');
const Person = require('./person');


// Display information from the StudentInfo module
console.log("Student Name: " + student.getName());
console.log("Campus Name: " + student.Location());
console.log("Date of Birth: " + student.dob);
console.log("Student Grade: " + student.StudentGrade(60));

// Create a new Person instance
const person1 = new Person("John", 30);
console.log(person1.displayInfo()); // Output: John is 30 years old.
```

Person.js code

person.js > person

```
class person {
    // Constructor to initialize name and age for each instance
    constructor(name, age) {
      this.name = name;
      this.age = age;
    }

    // Method to display person's info
    displayInfo() {
      return `${this.name} is ${this.age} years old.`;
    }
  }

  // Exporting the Person class so it can be used in other files
  module.exports = person;
```

Exercise 4 with output

```
const os = require("os");
const util = require("util");

//Display system information
console.log("Temporary directory: " + os.tmpdir());
console.log("Hostname: " + os.hostname());
console.log("OS: " + os.platform() + ", Release: " + os.release());
console.log("Uptime: " + (os.uptime() / 3600) + " hours");
console.log("User Info: " + util.inspect(os.userInfo()));
console.log("Total Memory: " + os.totalmem() / 1000000000 + " GB");
console.log("Free Memory: " + os.freemem() / 1000000000 + " GB");
console.log("CPU Info: " + util.inspect(os.cpus()));
console.log("Network Interfaces: " + util.inspect(os.networkInterfaces()));

// Program end
console.log("Program end");
```

Temporary directory: C:\Users\user\AppData\Local\Temp
Hostname: DESKTOP-4CFTA9B
OS: win32, Release: 10.0.22631
Uptime: 99.72650583333332 hours
User Info: {
  uid: -1,

```
  gid: -1,
  username: 'user',
  homedir: 'C:\\Users\\user',
  shell: null
}
Total Memory: 16.976740352 GB
Free Memory: 7.282839552 GB
CPU Info: [
  {
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
    times: {
      user: 2810421,
      nice: 0,
      sys: 1767671,
      idle: 57530218,
      irq: 351734
    }
  },
  {
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
    times: { user: 1733562, nice: 0, sys: 554968, idle: 59819484, irq: 30484
}
  },
  {
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
    times: {
      user: 3516437,
      nice: 0,
      sys: 1207906,
      idle: 57383640,
      irq: 41187
    }
  },
  {
```

```
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
    times: {
      user: 2172812,
      nice: 0,
      sys: 1045265,
      idle: 58889921,
      irq: 38390
    }
  },
  {
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
    times: {
      user: 3056812,
      nice: 0,
      sys: 1085562,
      idle: 57965625,
      irq: 35531
    }
  },
  {
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
    times: { user: 1809531, nice: 0, sys: 663328, idle: 59635125, irq: 15203
}
  },
  {
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
    times: { user: 2374671, nice: 0, sys: 739390, idle: 58993937, irq: 25281
}
  },
  {
    model: 'Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz',
    speed: 1992,
```

```
    times: { user: 2022656, nice: 0, sys: 652375, idle: 59432968, irq: 23000
}
  }
]
Network Interfaces: {
  WiFi: [
    {
      address: 'fe80::475b:88aa:b06a:e97',
      netmask: 'ffff:ffff:ffff:ffff::',
      family: 'IPv6',
      mac: '60:f2:62:f2:ca:60',
      internal: false,
      cidr: 'fe80::475b:88aa:b06a:e97/64',
      scopeid: 4
    },
    {
      address: '192.168.0.176',
      netmask: '255.255.255.0',
      family: 'IPv4',
      mac: '60:f2:62:f2:ca:60',
      internal: false,
      cidr: '192.168.0.176/24'
    }
  ],
  'Loopback Pseudo-Interface 1': [
    {
      address: '::1',
      netmask: 'ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff',
      family: 'IPv6',
      mac: '00:00:00:00:00:00',
      internal: true,
      cidr: '::1/128',
      scopeid: 0
    },
    {
      address: '127.0.0.1',
      netmask: '255.0.0.0',
```

```
    family: 'IPv4',
    mac: '00:00:00:00:00:00',
    internal: true,
    cidr: '127.0.0.1/8'
  }
 ]
}
Program end
```

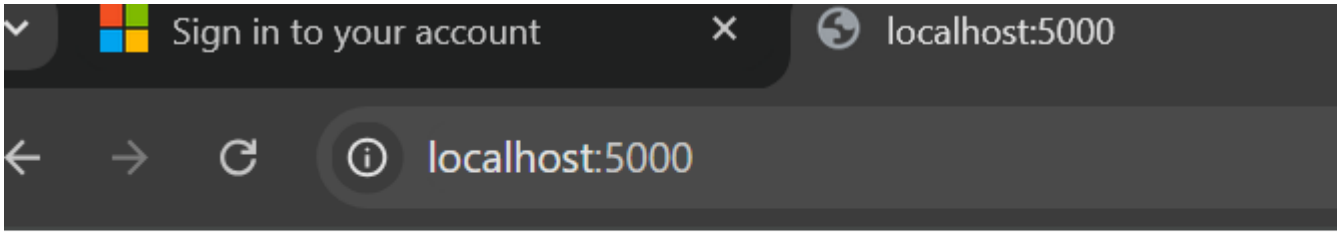
Week 3 reflective report

In today’s lab, I worked with Node.js and focused on creating and using modules. I started by defining a function called `EmployeeInfo` to display a welcome message with an employee's name and salary. Then, I learned how to create and use local modules in Node.js, such as `StudentInfo.js` and `Person.js`. I used the `require` function to import these modules into my main `index.js` file. The lab also covered how to export functions and variables from one file and use them in another. Additionally, I explored the `os` core module to gather system information, such as the operating system's platform, memory usage, and CPU details, and printed this data to the console. Overall, the lab helped me understand how to structure and organize my Node.js applications using modules, and how to interact with system information through core modules.

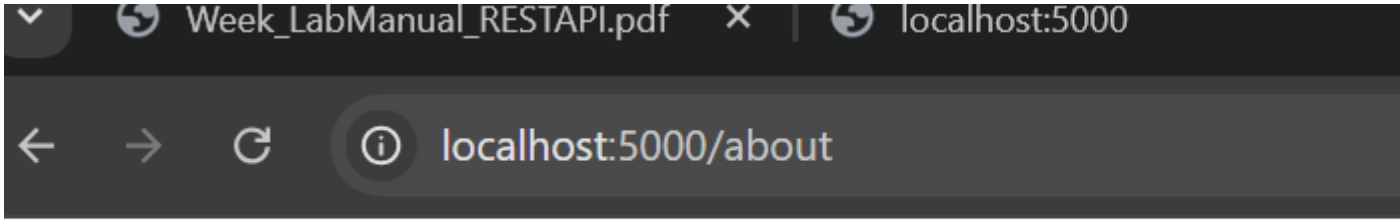
Week 4

C: > Users > user > Desktop > JS node.js > ...

```
let express = require("express")
let fs =require("fs")
let app=express()
let bodyParser= require("body-parser")

app.use(bodyParser.urlencoded({extended:true}));


app.post()
app.put()
//res=response
//req=request
//npm init-init stands for initialise    "initialise": Unknown w
app.get('/',function(req,res){
    res.send("hello it is my frist express application ")})    '

    app.get("/users/:userid/books/:bookid",function(req,res){
        res.send(req.params)
    })
    app.listen(5000,function(){
        console.log("server is running on port 5000")})

```



ello it is my first express application



This is basic express application

labsession

Welcome | index.js | student.json | package.jso

index.js > app.get('/GetStudentid/:id') callback

```js
app.get('/GetStudentid/:id', (req, res) => {    "Stu
    const students = {
        "Student1": { "name": "Alice", "age": 21 },
        "Student2": { "name": "Bob", "age": 22 },
        "Student3": { "name": "Charlie", "age": 23 },
        "Student4": { "name": "David", "age": 24 }
    };

    const student = students["Student" + req.params.
    if (student) {
        res.json(student);
    } else {
        res.json({
            status: true,
            Status_Code: 200,
```

PROBLEMS 11 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\user\Desktop\labsession> node index.js
server is running on port 5000
student { name: 'Alice', age: 21 }
student { name: 'Bob', age: 22 }
student { name: 'Charlie', age: 23 }
student { name: 'Bob', age: 22 }
student { name: 'David', age: 24 }
```

Week_LabManual_RESTAPI.pdf × localhost:5000

localhost:5000/GetStudentid/1

retty-print

"name":"Alice","age":21}

Week_LabManual_RESTAPI.pdf × localhost:5000/GetStudenti

localhost:5000/GetStudentid/4

retty-print

"name":"David","age":24}

Week_LabManual_RESTA × localhost:5000 ×

localhost:5000/GetStudentid/2

etty-print

name":"Bob","age":22}

Week_LabManual_RESTAPI.pdf × localhost:5000/GetStudentid/3

localhost:5000/GetStudentid/3

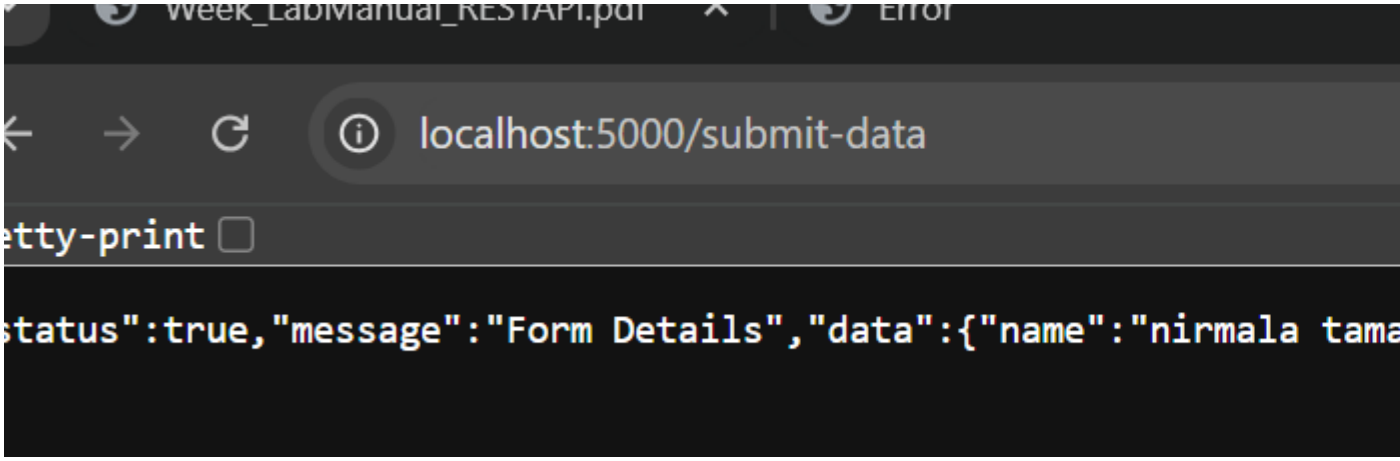
tty-print

ame":"Charlie","age":23}

Welcome | index.js | student.json | package.json 2

index.js > app.get('/GetStudentid/:id') callback

```javascript
var express = require("express");
var app = express();

// Middleware function for body parsing
var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: true }));

// Default route
app.get('/', function(req, res) {
    res.send("Hello, this is my first Express application!");
});

// Start the server
app.listen(5000, function() {
    console.log("Server is running on port 5000");
});

// About route
app.get('/about', function(req, res) {
    res.send("This is a basic Express application");
});

// Users route
app.get('/users/:userId/books/:bookId', function(req, res) {
    res.send(req.params);
});

// Updated /GetStudentid/:id route without JSON file    "Studentid": Unknown word.
app.get('/GetStudentid/:id', (req, res) => {    "Studentid": Unknown word.

    const students = {
        "Student1": { "name": "Alice", "age": 21 },
        "Student2": { "name": "Bob", "age": 22 },
        "Student3": { "name": "Charlie", "age": 23 },
        "Student4": { "name": "David", "age": 24 }
    };

    const student = students["Student" + req.params.id];
    if (student) {
        res.json(student);
    } else {
        res.json({
            status: true,
            Status_Code: 200,
            "Requested At": new Date().toISOString(),
            "Request URL": req.url,
            "Request Method": req.method,
            studentData: "Student not found"
        });
    }
```


Week_LabManual_RESTAPI.pdf
Error
localhost:5000/submit-data
etty-print
status":true,"message":"Form Details","data":{"name":"nirmala tama

Student Details

First Name:

Last Name :

Email:

Age :

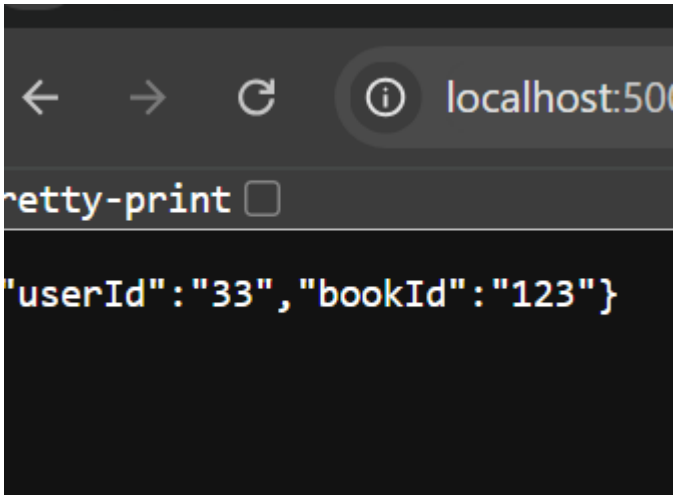
Please select your gender:

- ○ **Male**
- ○ **Female**
- ○ **Other**

Qualifications

- ☐ **GCSE**
- ☐ **A- level**
- ☐ **Higher National Certificate/Level 4**
- ☐ **Foundation Degree/HND/DipHE/Level 5**
- ☐ **Bachelor Degree/Graduate diploma or Certificate/Level 6**
- ☐ **Master Degree/PGCE/Level7**
- ☐ **PhD/Level8**

Submit



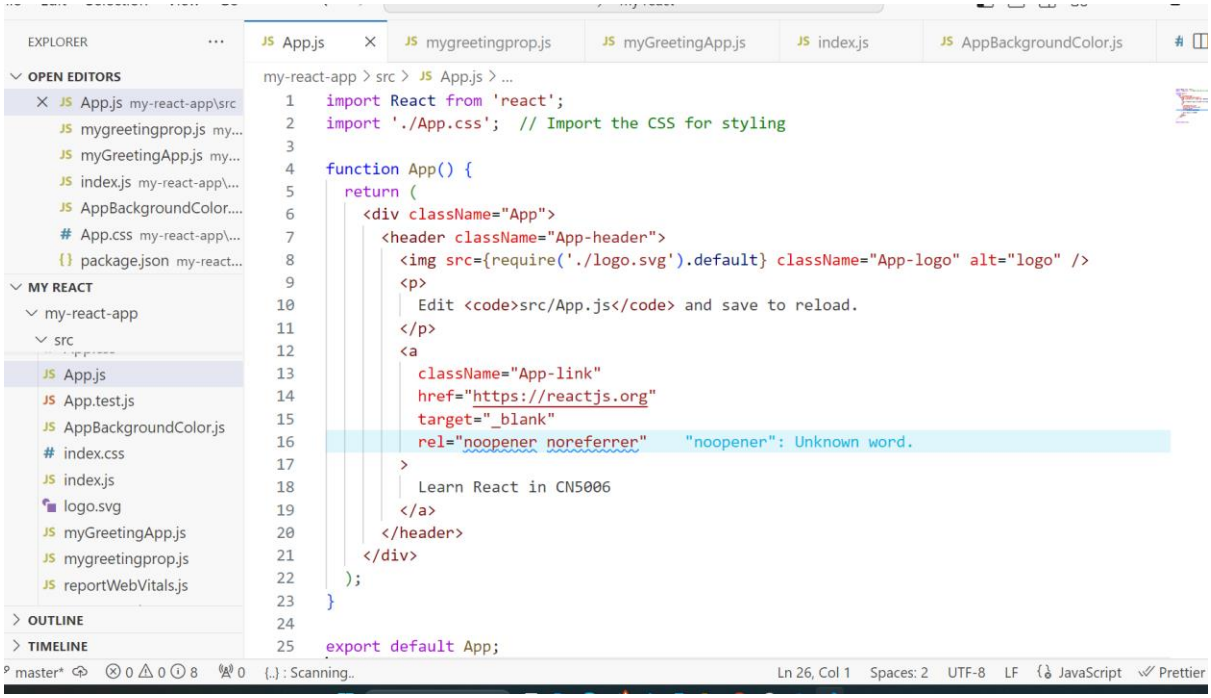
{"status":true,"message":"form Details","data":{"name":"SoneOne Johnson ","age":"33 Gender: male ","Qualification":" QualificationGCSE,PhD"}}

= Reflective report of week 4.

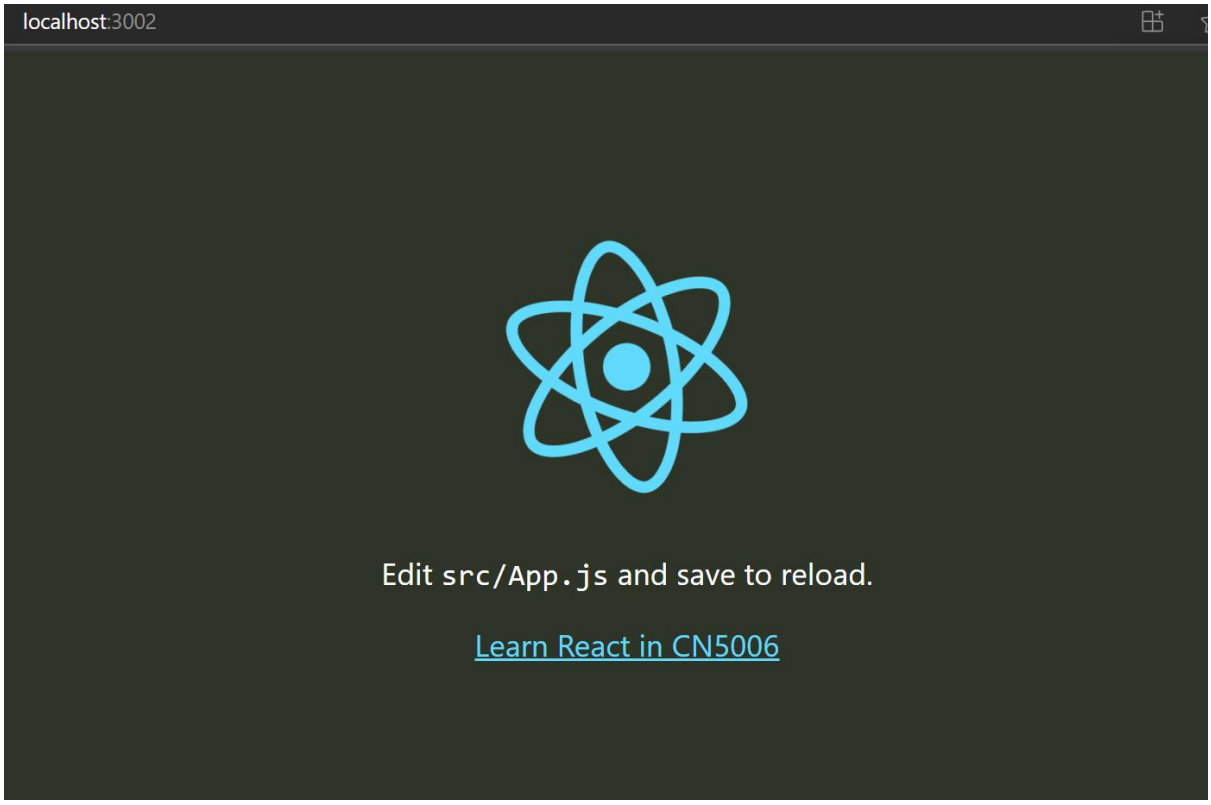
In this lab session, I learned how to set up a basic Node.js server using Express and handle form submissions from an HTML page. I worked on creating an Express server to serve static files and process form data sent via POST requests. The main challenge was handling checkbox data correctly by ensuring it was an array before applying methods like `.join()`. I also encountered issues like port conflicts, which were resolved by either changing the port or killing existing processes. This session helped me understand how to process user input from forms, send JSON responses, and debug common errors. Overall, it was a valuable experience that strengthened my skills in server-side development with Node.js and Express.

Week 5 TASK 1

Visual Studio to begin working with your React application with changed text

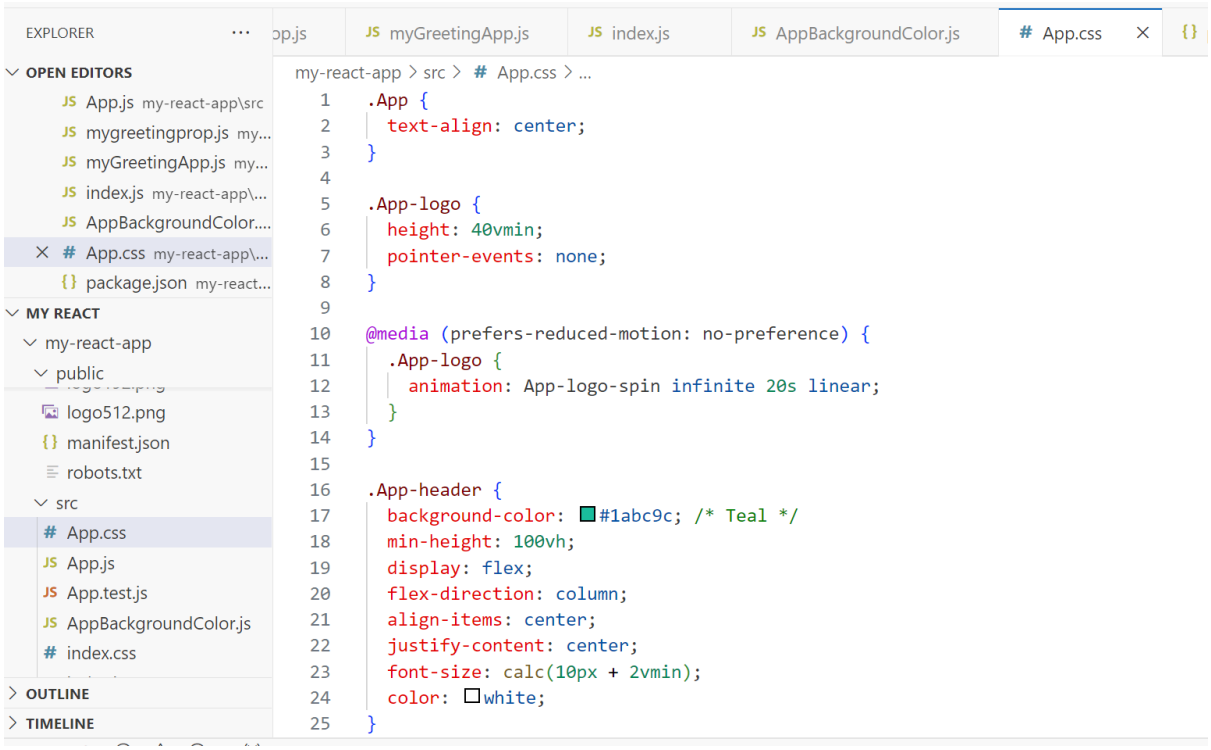



```
PS C:\Users\user\Desktop\my react\my-react-app> npm start

> my-react-app@0.1.0 start
> react-scripts start
? Something is already running on port 3000.
√ Something is already running on port 3000.

Would you like to run the app on another port instead? ... yes
(node:12144) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterS
etupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:12144) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBefor
Starting the development server...
Compiled successfully!

You can now view my-react-app in the browser.

  Local:            http://localhost:3002
  On Your Network:  http://10.210.85.219:3002

Note that the development build is not optimized.
```

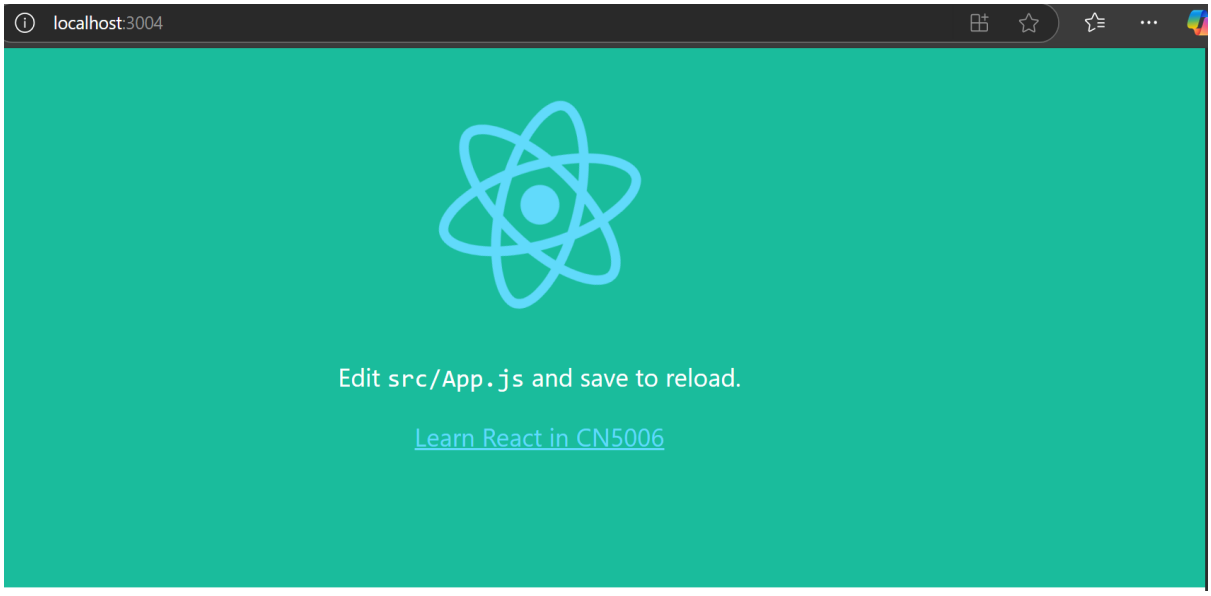


TASK 2

Task2: Change the background colour of your application:



```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SPELL CHEC

√ Something is already running on port 3000.

Compiled successfully!

You can now view my-react-app in the browser.

  Local:            http://localhost:3004
  On Your Network:  http://192.168.0.176:3004

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Task3 : Creating stateless function component using React

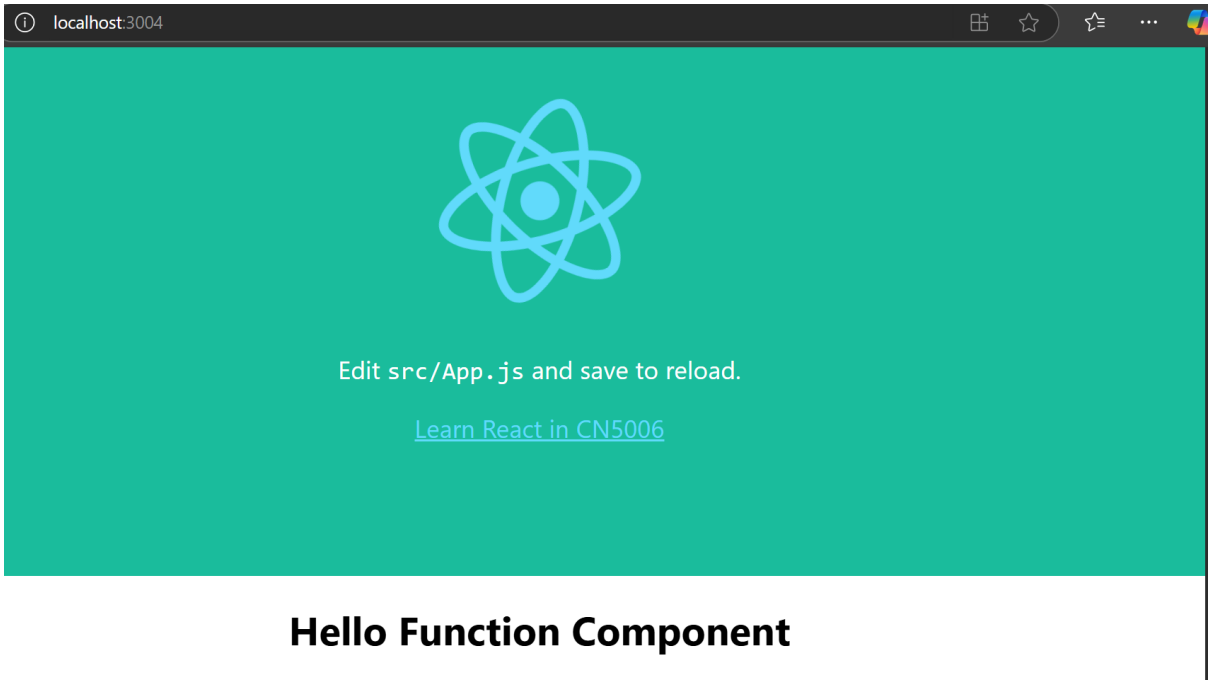
App.js | myGreetingApp.js | index.js | App.css | package.json

my-react-app > src > myGreetingApp.js > ...

```javascript
// Import App.css for styles
import './App.css';

// Define a functional component
function GreetingElement() {
  // Define a variable to hold the greeting message
  const greeting = "Hello Function Component";

  // Return JSX
  return (
    <div className="App">
      <h1>{greeting}</h1>
    </div>
  );
}

// Export the component so it can be used in other files
export default GreetingElement;
```

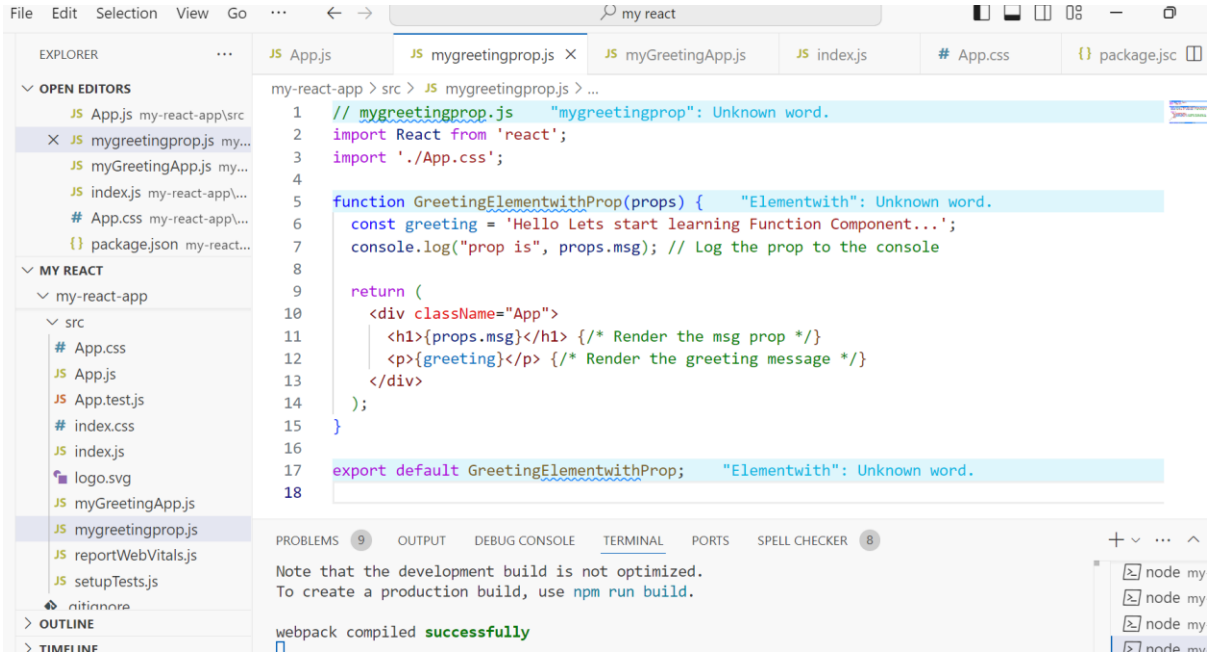
PROBLEMS 3 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | SPELL CHEC

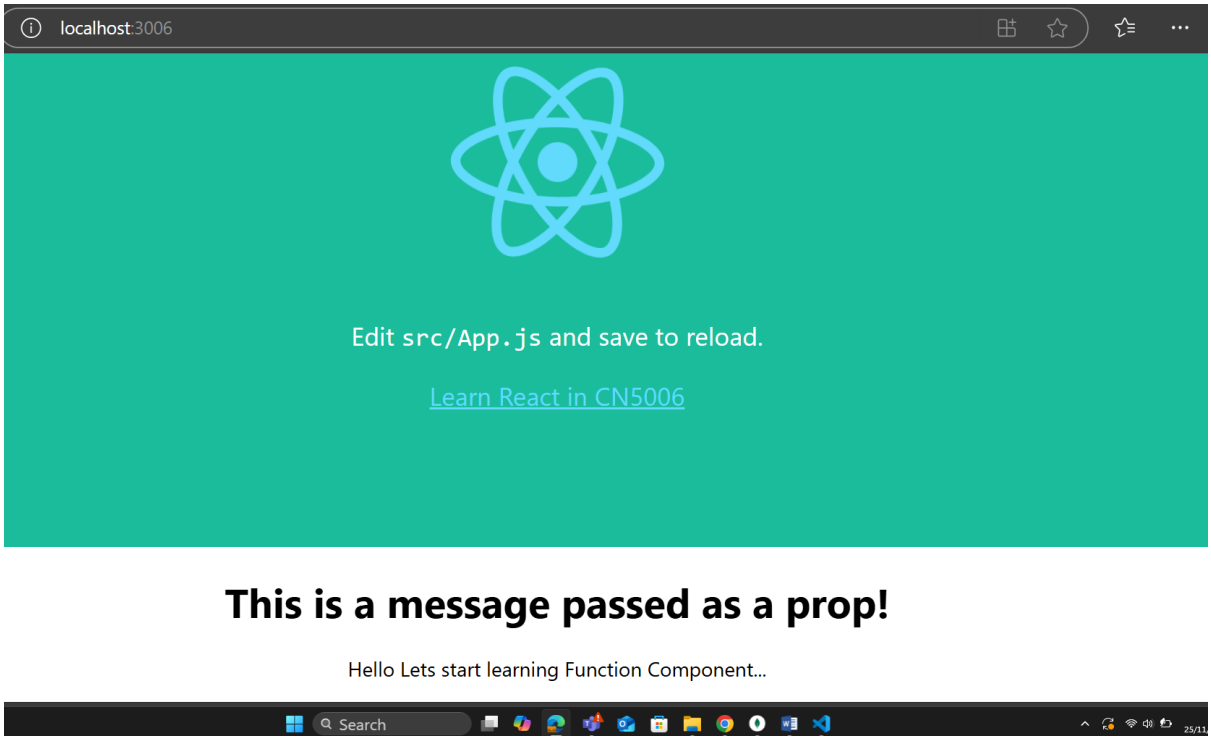
```
√ Something is already running on port 3000.

Compiled successfully!

You can now view my-react-app in the browser.

  Local:            http://localhost:3004
  On Your Network:  http://192.168.0.176:3004

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



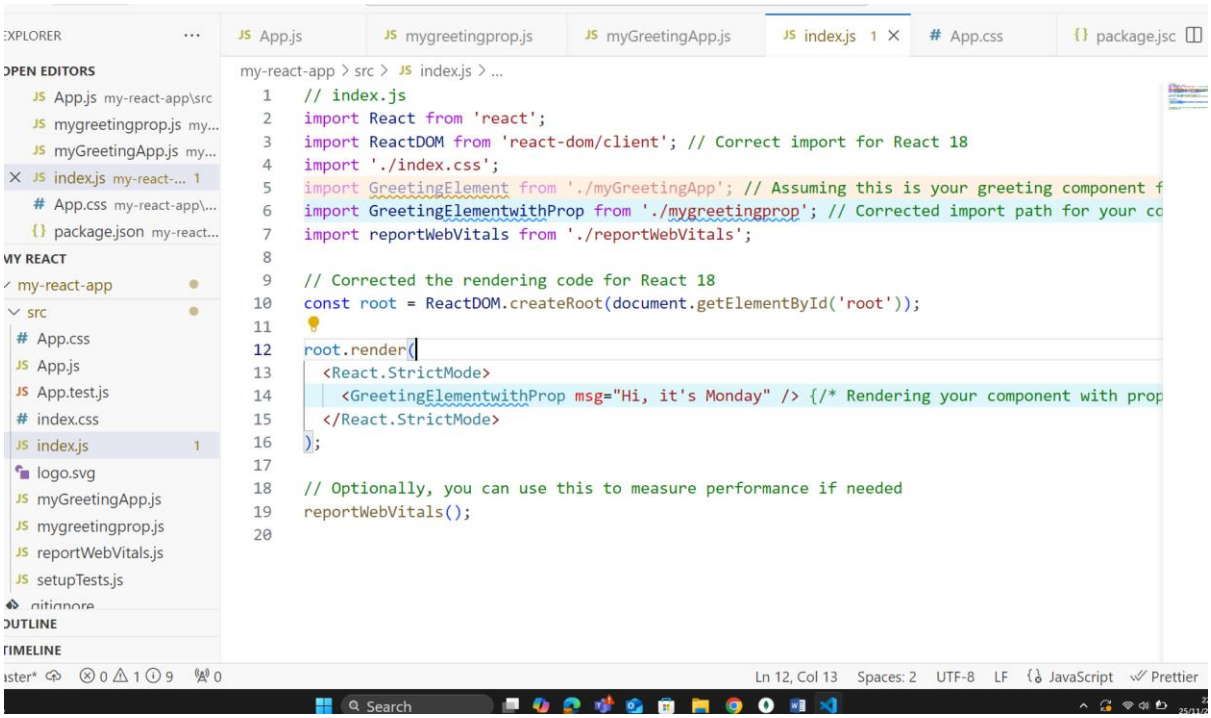
Hello Function Component

Task 4 Creating the Functional Component using properties





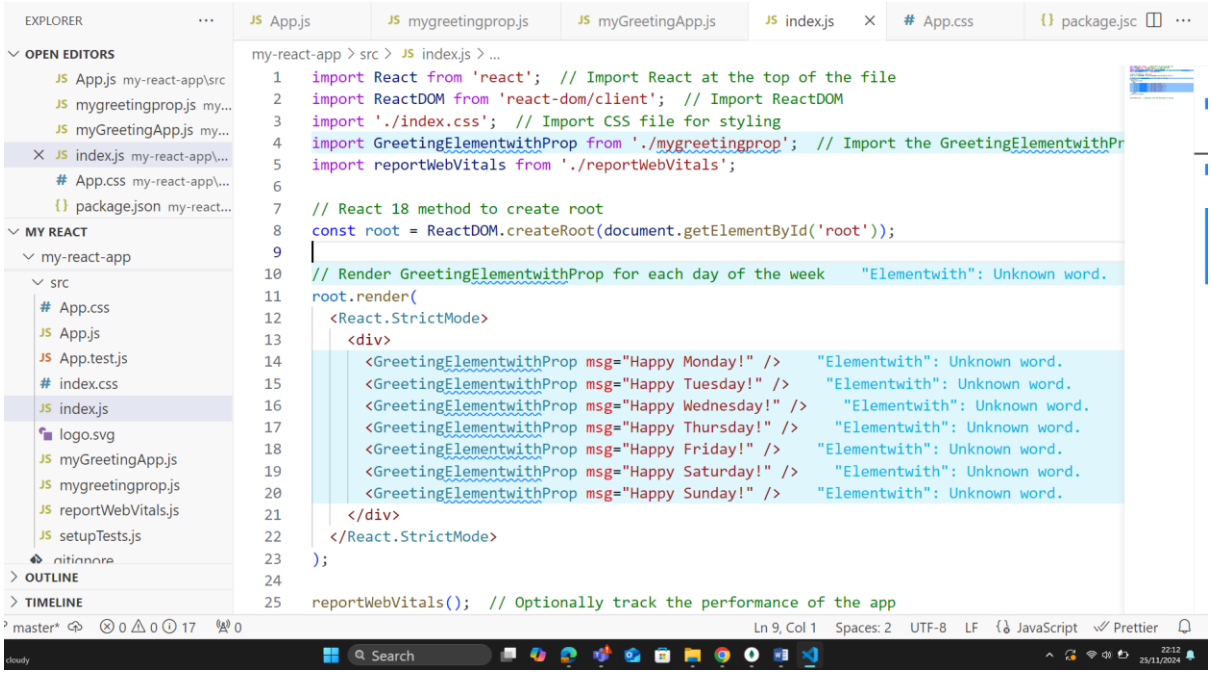
Step 2: Now in index file import the file Mygreetingprops.js using the line import GreetingElementwithProp from './myGreetingProp'; nrxt, clear the line that we used in Task3 and inplace of this create a new element that we have defined inmyGreetingProp.js. use following line to do it:

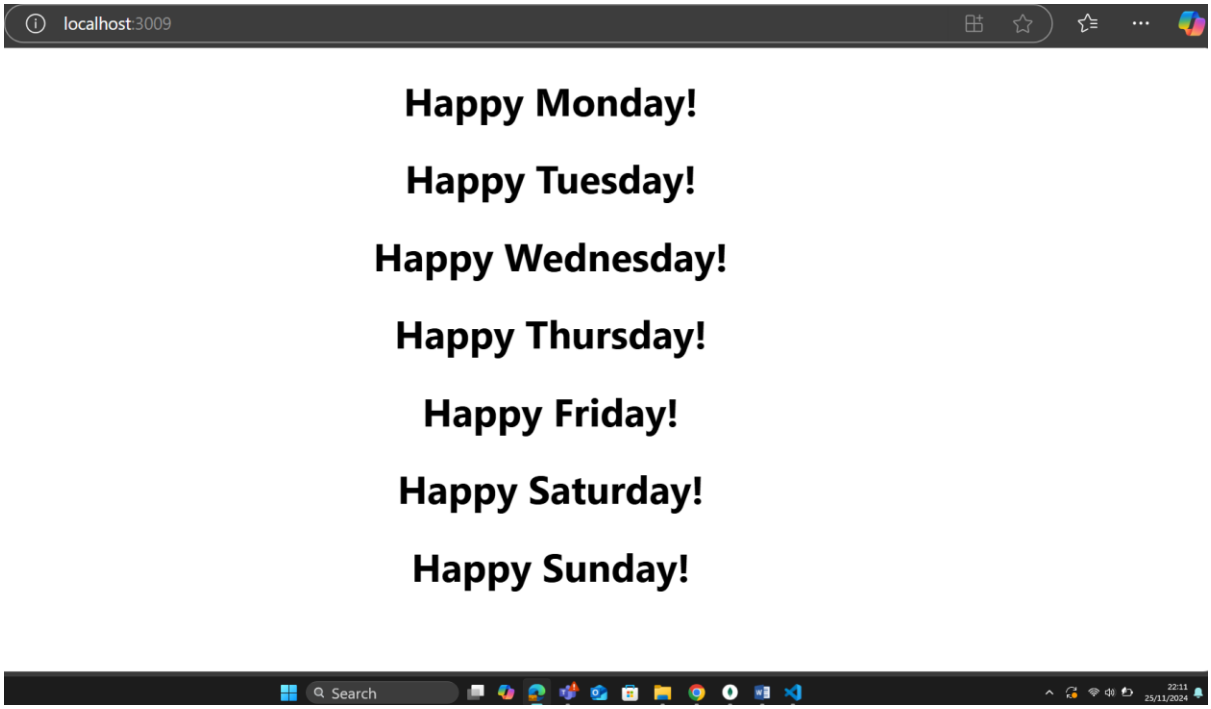


Hi, it's Monday

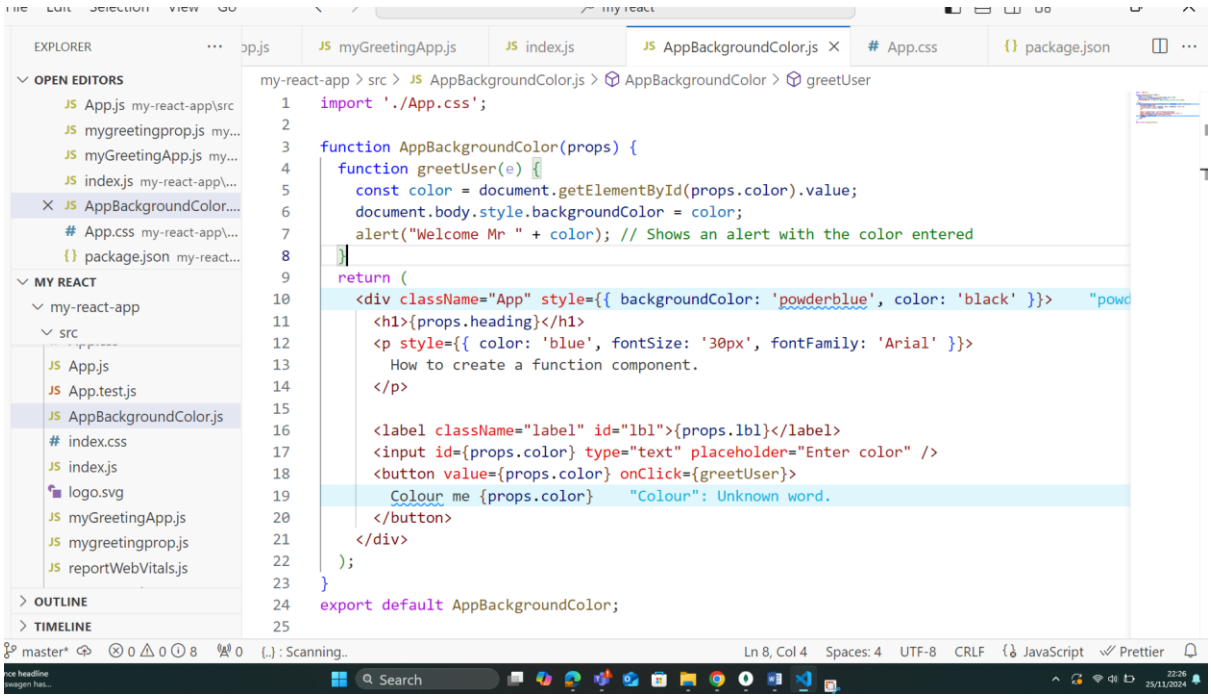
Hello Lets start learning Function Component...

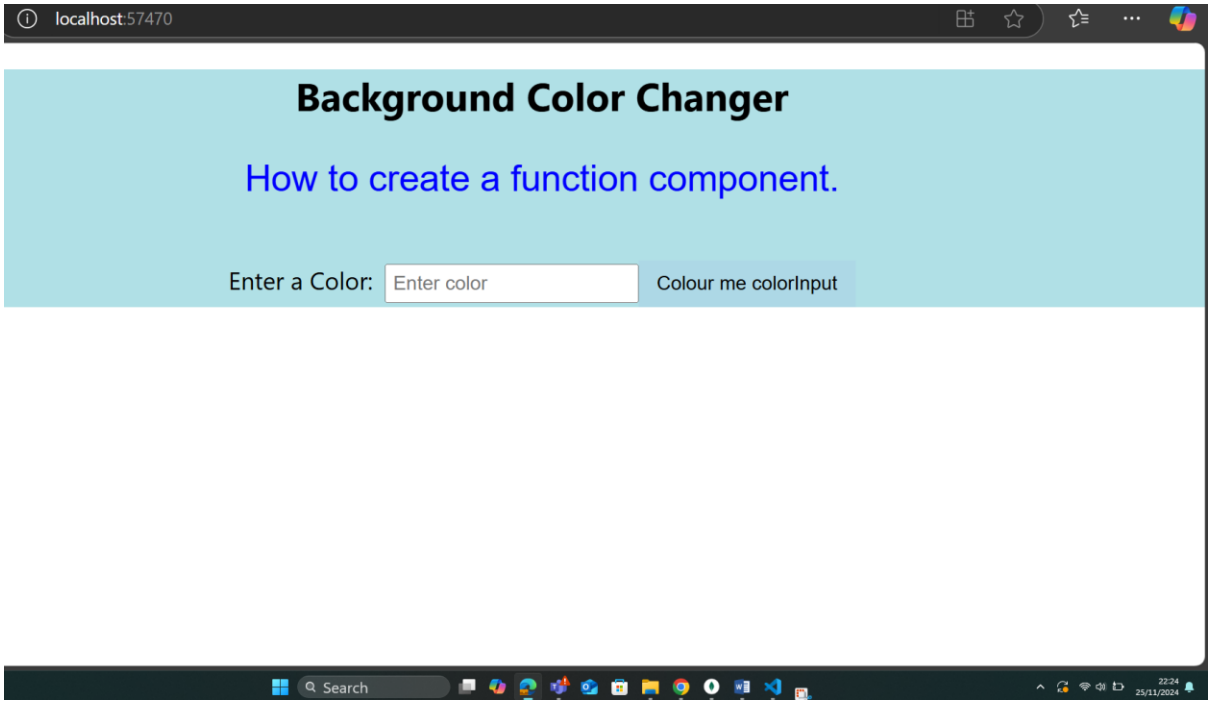
Task 5 . Use the same GreetingElementwithProp to display seven days of week greeting message. Hint use

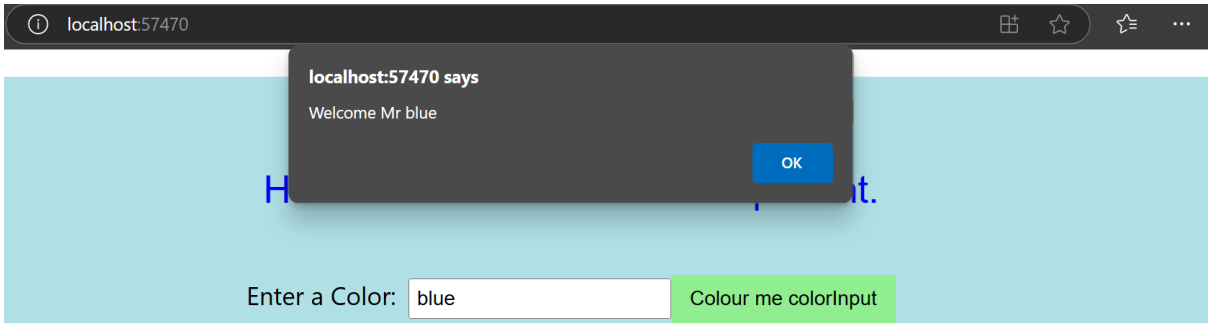




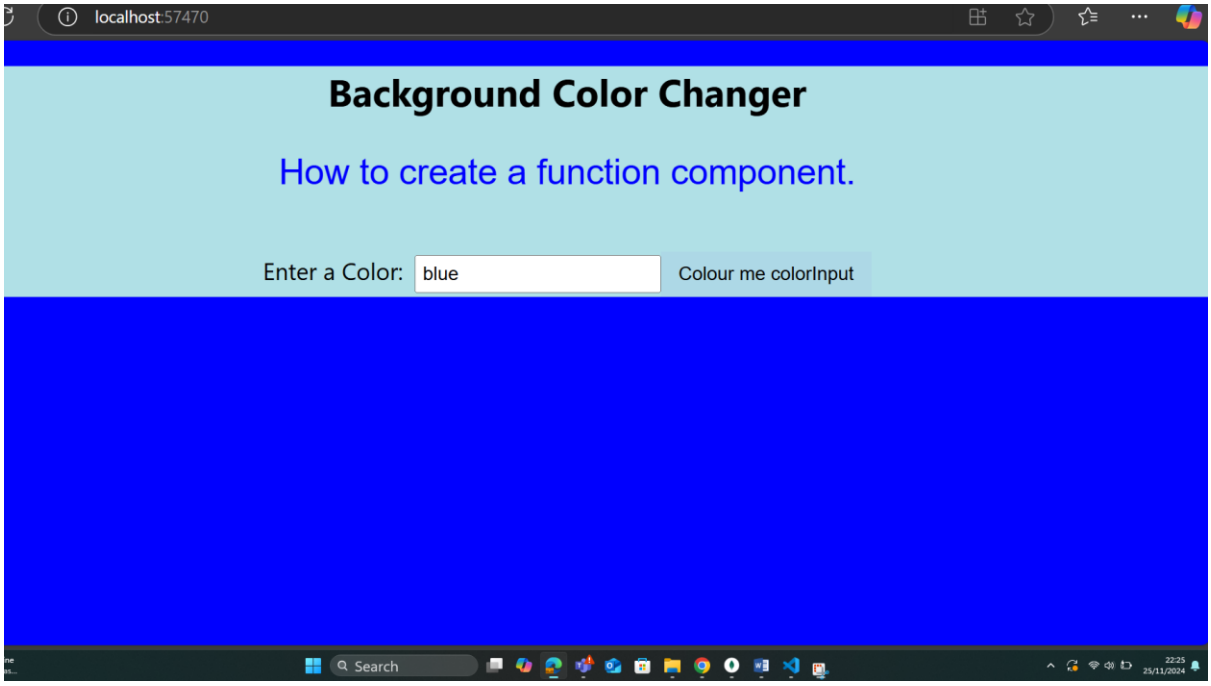
Task6 : Functional Components using properties and HTML elements such as buttons In this task we will be using html tag button ,text and labels in our functional component. The idea is that we want to create a React component that can change the background colour on the click of the button. The methodology is still the same. We create this component in separate js file, name it as AppbackgroundColor.js . Add the code as given in the black window




localhost:57470
Background Color Changer
How to create a function component.
Enter a Color:
Enter color
Colour me colorInput
Search


localhost:57470
localhost:57470 says
Welcome Mr blue
OK
Enter a Color:
blue
Colour me colorInput


Search



For todays Lab submission, After you complete the lab write down a word document answering following questions for your portfolio

: 1. What is React?

= React is a tool for building fast, interactive websites by creating reusable components. It updates only the parts of the page that change, making it efficient and smooth.

2. What do you understand by React component and what command do you use to create a React component with or without property?

= A React component is a function or class that takes inputs (called "props") and returns how the UI should look. It helps create complex UIs by combining simple, reusable pieces.

- **Without property (stateless functional component)**:
  ```
  JAVASCRIPT
  function MyComponent() {
   return <div>Hello, World!</div>;
    }
  ```
- **With property (props-based component)**:
  ```
  Javascript

  function MyComponent(props) {
  ```

```
      return <div>{props.message}</div>;
        }
```

3. What command will you use to render the the newly created component named as MyReact?

= render a newly created component called `MyReact`, we would typically use the `ReactDOM.render()` method in the `index.js` file. Assuming we are using a function component:

```
javascript

import React from 'react';
import ReactDOM from 'react-dom';
import MyReact from './MyReact'; // Assuming MyReact
component is defined in MyReact.js

ReactDOM.render(<MyReact />,
document.getElementById('root'));
```

4. Suppose the MyReact Component has a property heading, write down the code that could be used to render the MYReact Component, and pass the message to the property heading as "this is my first element"?

```
= // MyReact.js
function MyReact(props) {
  return <h1>{props.heading}</h1>;
}

// In your main render file (usually index.js or
App.js)
ReactDOM.render(<MyReact heading="This is my first
element" />, document.getElementById('root'));.
```

5 What is the name of the React Component b. How many properties this component uses?

```
= <AppColor heading="This is first element"
lbl="Name :" color="green" />
```

6num)

a. What is the name of the React Component ?
   = The name of the React component is `AppColor`.
b. How many properties this component uses?

= The component `AppColor` uses **3 properties** (also known as **props**):

- heading
- lbl
- color

7） Look at the following Code:

function GreetingElementwithProp(props) { return (

Wellcome , {props.studentname}

;

); } export default ?????? what will you write to make this export this function correctly? Hint you need to replace ?????? with the correct word.

= function GreetingElementwithProp(props) {

```
  return (
    <div className="App">
      <h1>Welcome, {props.studentname}</h1>
    </div>
  );
}

export default GreetingElementwithProp;
```

Add a function that takes two properties as numbers ,add these numbers on the click event of the button and display the sum.?

= import React, { useState } from 'react';

```
function AddNumbers(props) {
  const [sum, setSum] = useState(0);

  const addNumbers = () => {
    setSum(props.number1 + props.number2);
  };

  return (
    <div className="App">
      <h1>The sum is: {sum}</h1>
      <button value={props.color} onClick={addNumbers}>Add
Numbers</button>
    </div>
  );
}

export default AddNumbers;
```