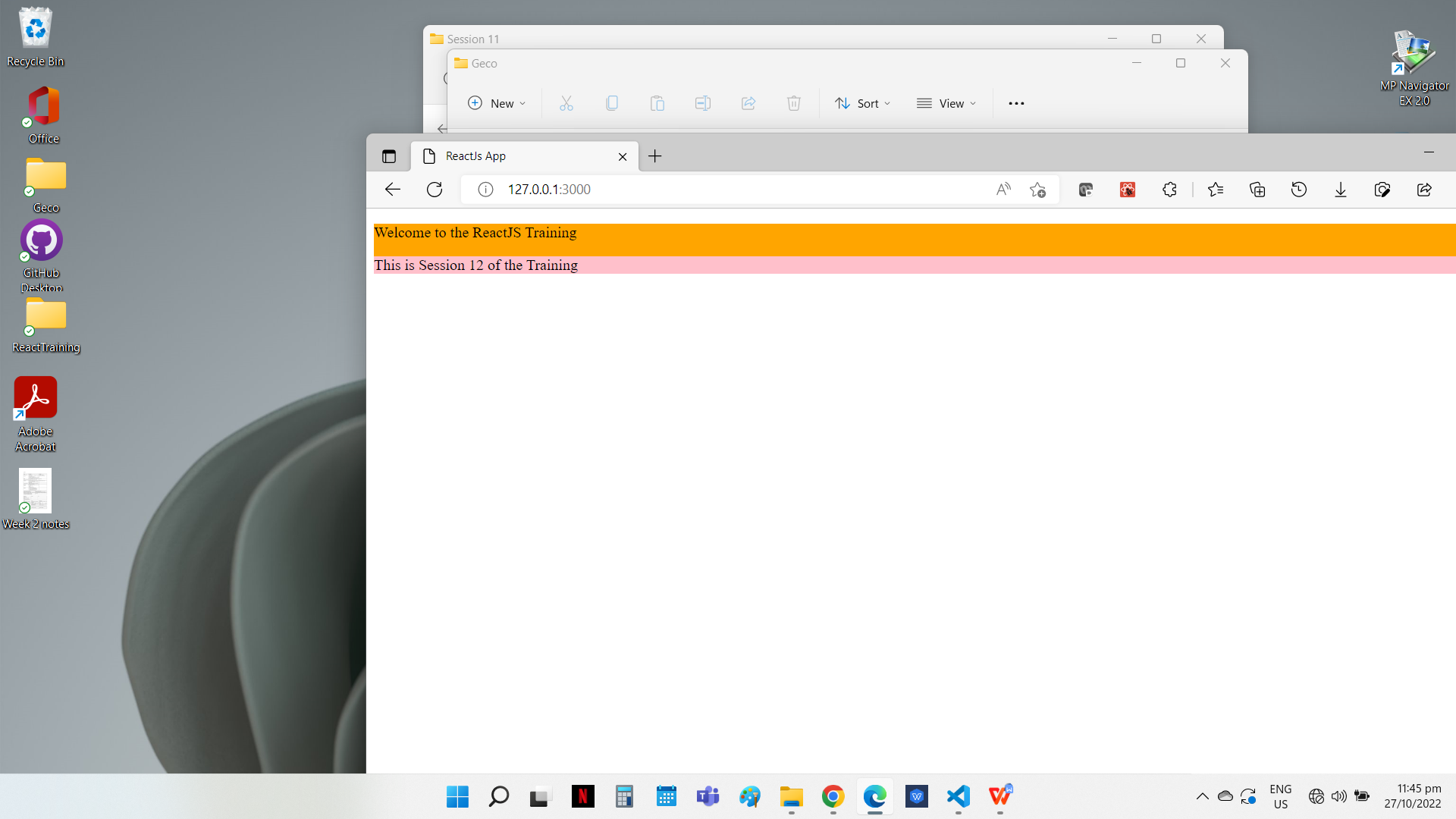
**Geco ReactJS Training**

|  |  |
| --- | --- |
| ReactJS Assignment |  |
| Name: | Ho Choo Geok |
| Date of Issue: | 26th October 2022 |
| Date of Submission: | 30th October 2022 |
| Cohort No.: | 27 |
| Week No.: | 3 |
| Session No.: | 12 |

1. Create a Hoc that add random color to background of 2 component.

Solution: Screen capture of the output as shown below.



**Source Code**

File Name: App.js

import React from "react";

import HOC from './HOC';

import Hook from "./Hook";

const App = () => {

  return (

    <>

    <p >Welcome to the ReactJS Training </p>

    <Hook />

    </>

)

};

export default HOC(App);

File Name: HOC.js

import React from 'react';

const HOC = (Propspect) => {

    const color = [ 'pink','blue','orange' ];

    const randomColor = color [Math.floor(Math.random()\*4)];

    return() => {

        return(

            <div>

                <p style={{backgroundColor:randomColor}}> <Propspect /> </p>

            </div>

        )

    }

} ;

export default HOC;

File Name: Hooks.js

import React from 'react';

import HOC from './HOC';

const Hooks = () => {

    return(

        <>

        <p> This is Session 12 of the Training </p>

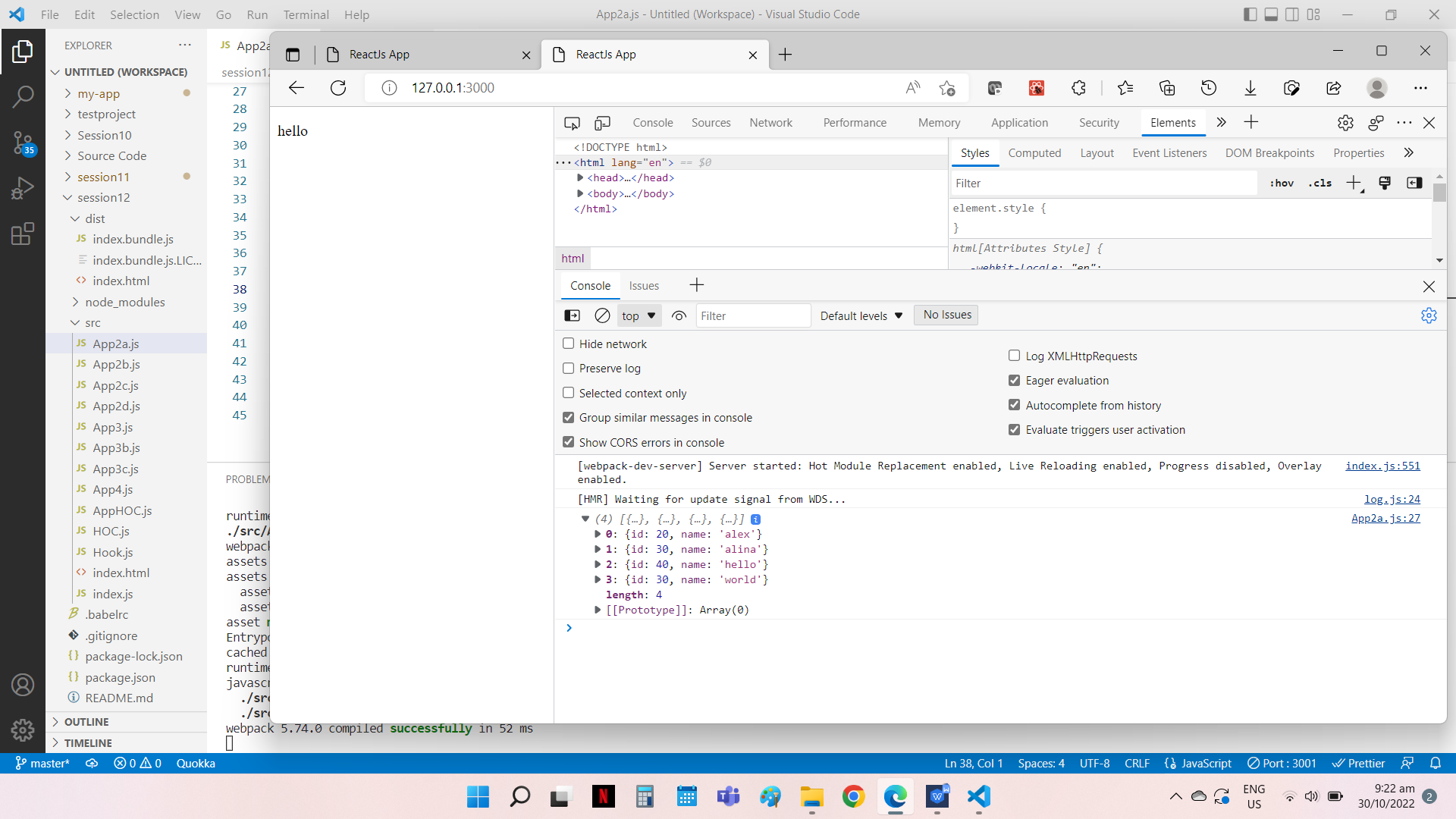
        </>

    )

};

export default HOC(Hooks);

1. Advance task (use lodash.js)
2. symmetric difference of 2 arrays

Source Code

File Name: App2a.js

import React from 'react';

import \_ from "lodash";

const App = () => {

const arrOne = [

        {

        id: 20,

        name: 'alex'

        },

        {

        id: 30,

        name: 'alina'

        }

    ]

const arrTwo = [

    {

        id: 40,

        name: 'hello'

    },

    {

        id: 30,

        name: 'world'

    }

]

let test= \_.xor(arrOne,arrTwo);

console.log(test);

//Trying out different methods and get the same outcome

// let diff1 = \_.difference(arrOne, arrTwo);

// console.log(diff1);

// let diff2 = \_.difference(arrTwo,arrOne);

// console.log(diff2);

// let symmetric=\_.union(diff1,diff2);

// console.log(symmetric);

    return(

        <div>

            <p> hello</p>

        </div>

    );

};

export default App;

2b) const str = ['u', 'ec']

const arr = [{

    storageVal: 'u',

     table: ['E', 'F']

},{

    storageVal: 'data',

    id: 3

}, {

    storageVal: 'ec',

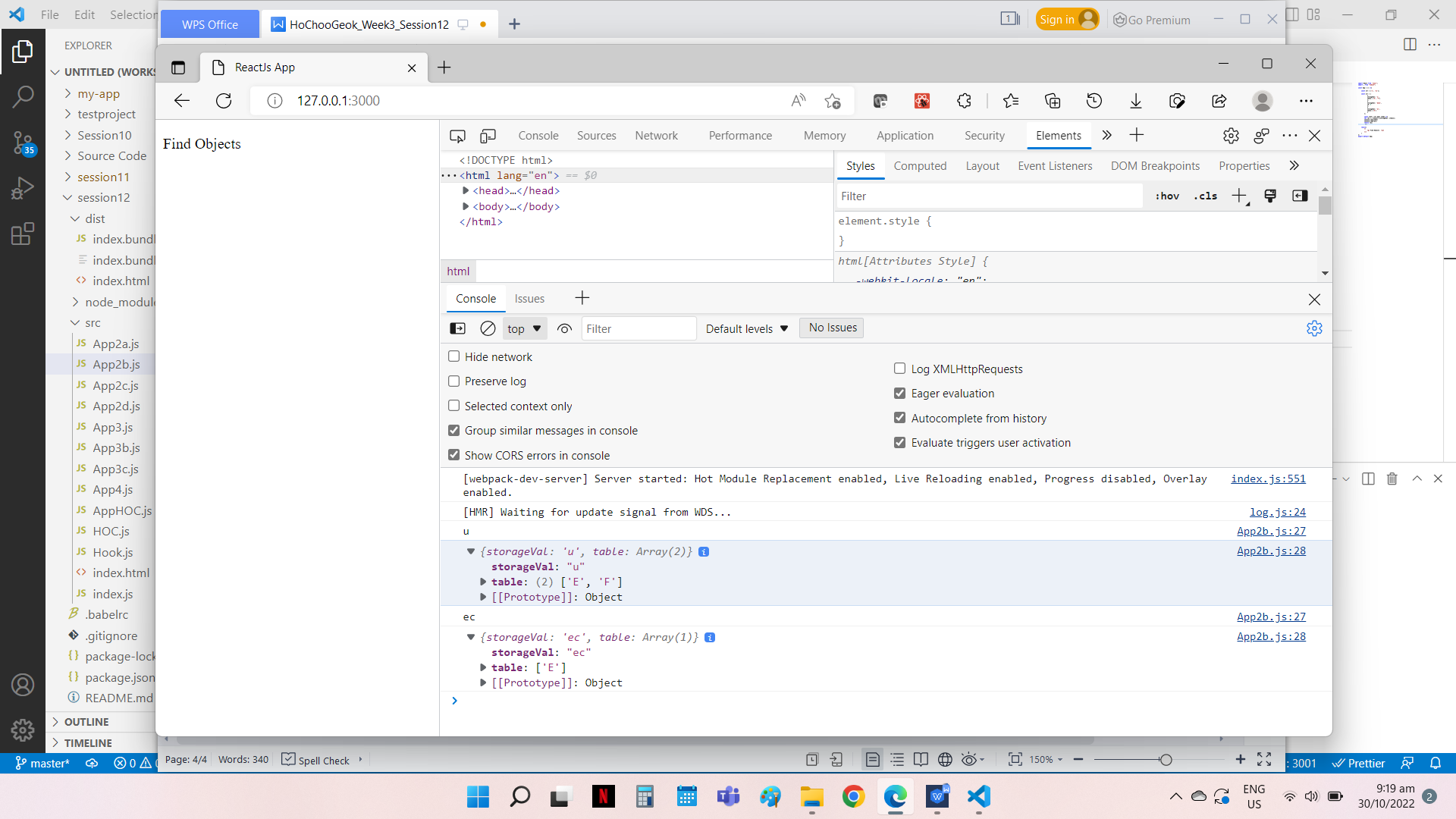
    table: ['E']

}]

get all the tables ['E', 'F'], ['E']

Source Code

File Name: App2b.js

import React from 'react';

import \_ from "lodash";

const App = () =>{

    const str = ['u', 'ec'];

    const arr = [

            {

            storageVal: 'u',

            table: ['E', 'F']

            },

            {

            storageVal: 'data',

            id: 3

            },

            {

            storageVal: 'ec',

            table: ['E']

            }

        ];

        const item = str.map( items =>{

        let res =\_.find(arr,{storageVal: items});

        console.log(items)

        console.log(res);

        return res;

         });

    return(

        <>

            <p> Find Objects  </p>

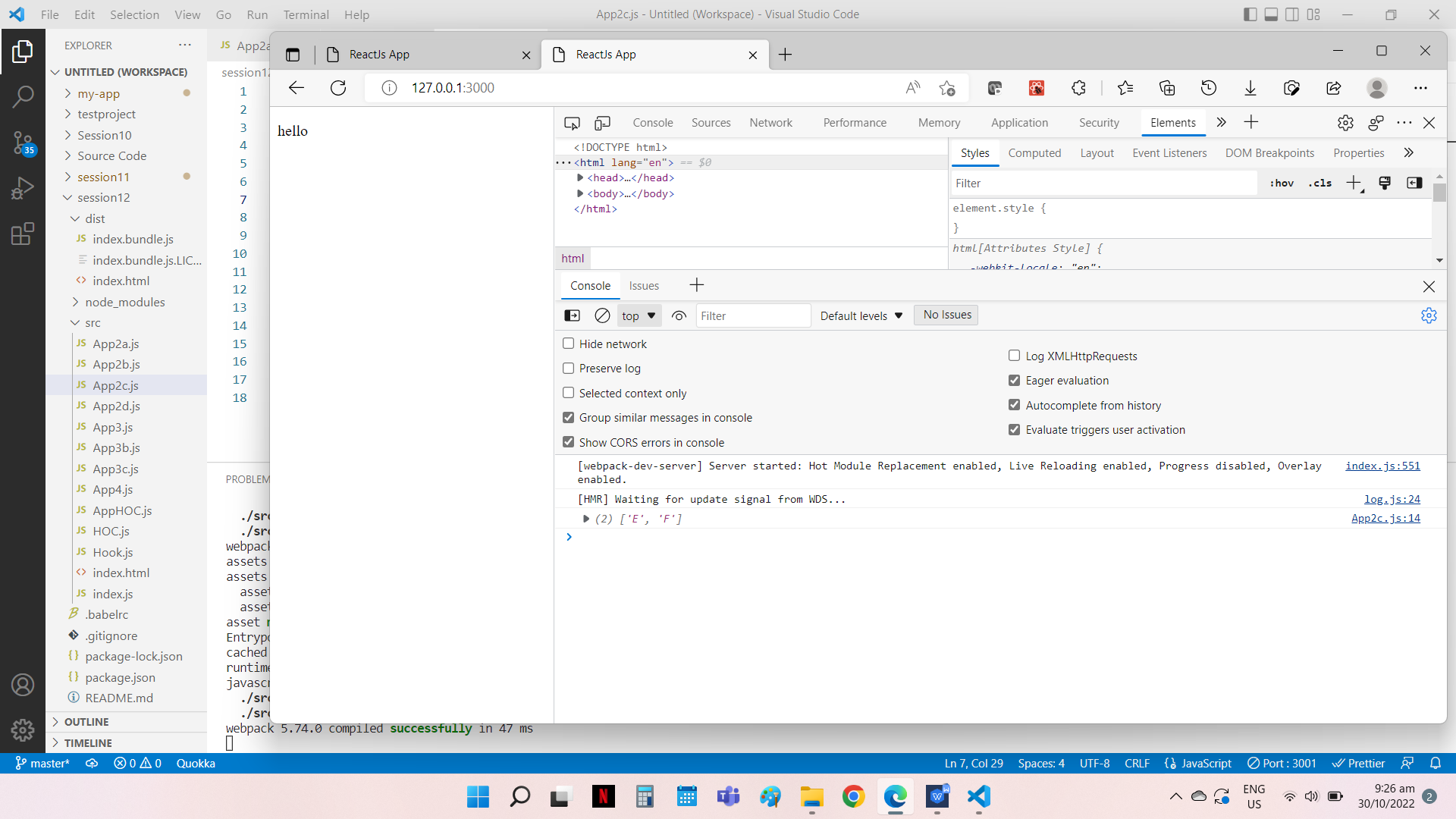
        </>

    )

};

export default App;

2c) Flatten an array const a = [['E'], ['F']]

Source Code

File Name: App2c.js

import React from 'react';

import \_ from 'lodash';

function App(){

    const a = [['E'], ['F']]

    let newarr=\_.flatten(a);

    console.log(newarr);

    return(

        <>

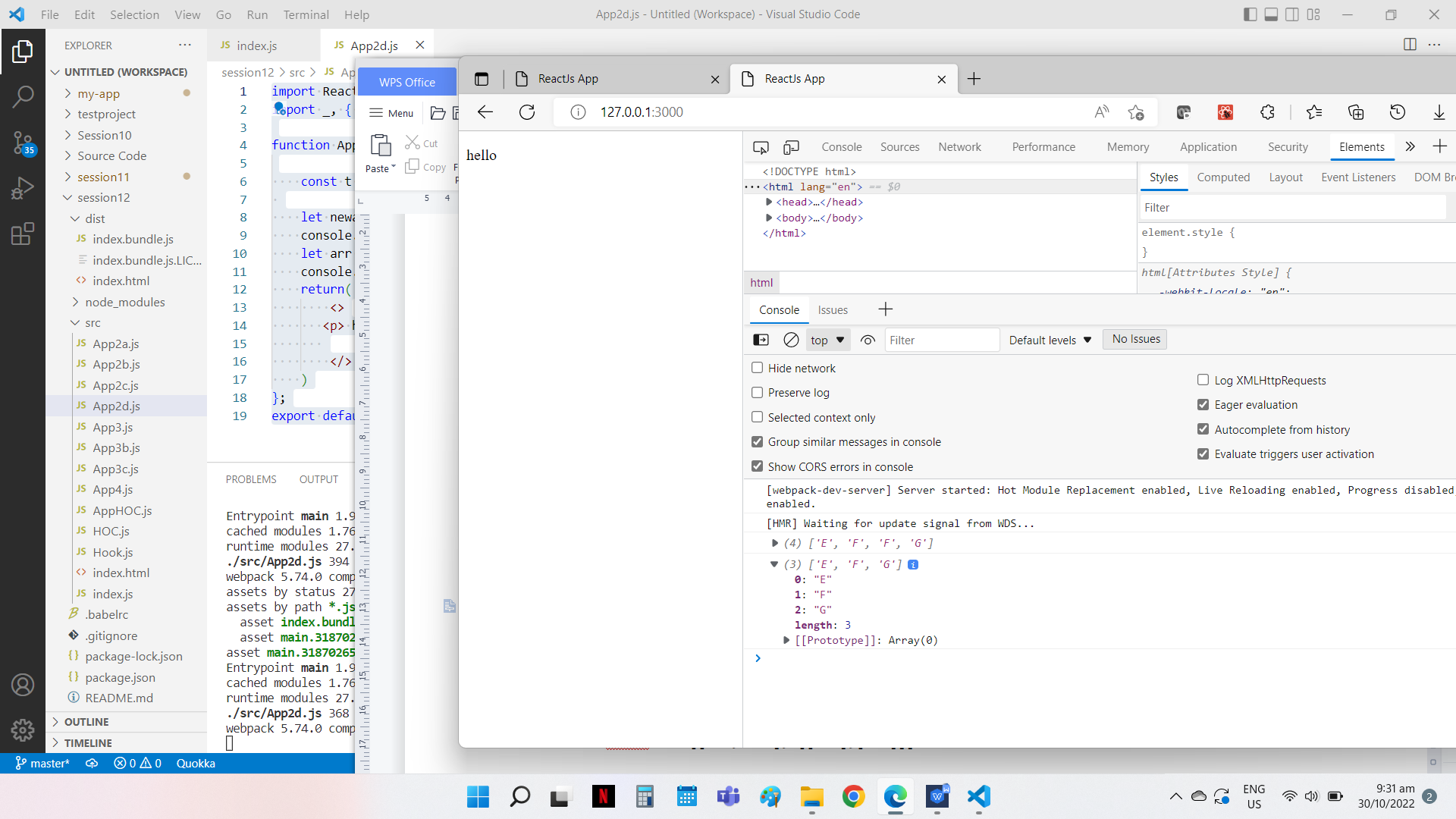
       <p> hello</p>

      </>

    )

};

export default App;



2d) const t = ['E', 'F'], [['F'], ['G']]

Source Code

File Name: App2d.js

import React from 'react';

import \_, { set } from 'lodash';

function App(){

    const t = [['E', 'F'], [['F'],['G']]]

     let newarr=\_.flattenDepth(t,2);

    console.log(newarr);

    let arr = \_.uniq(newarr);

    console.log(arr);

    return(

        <>

       <p> hello</p>

        </>

    )

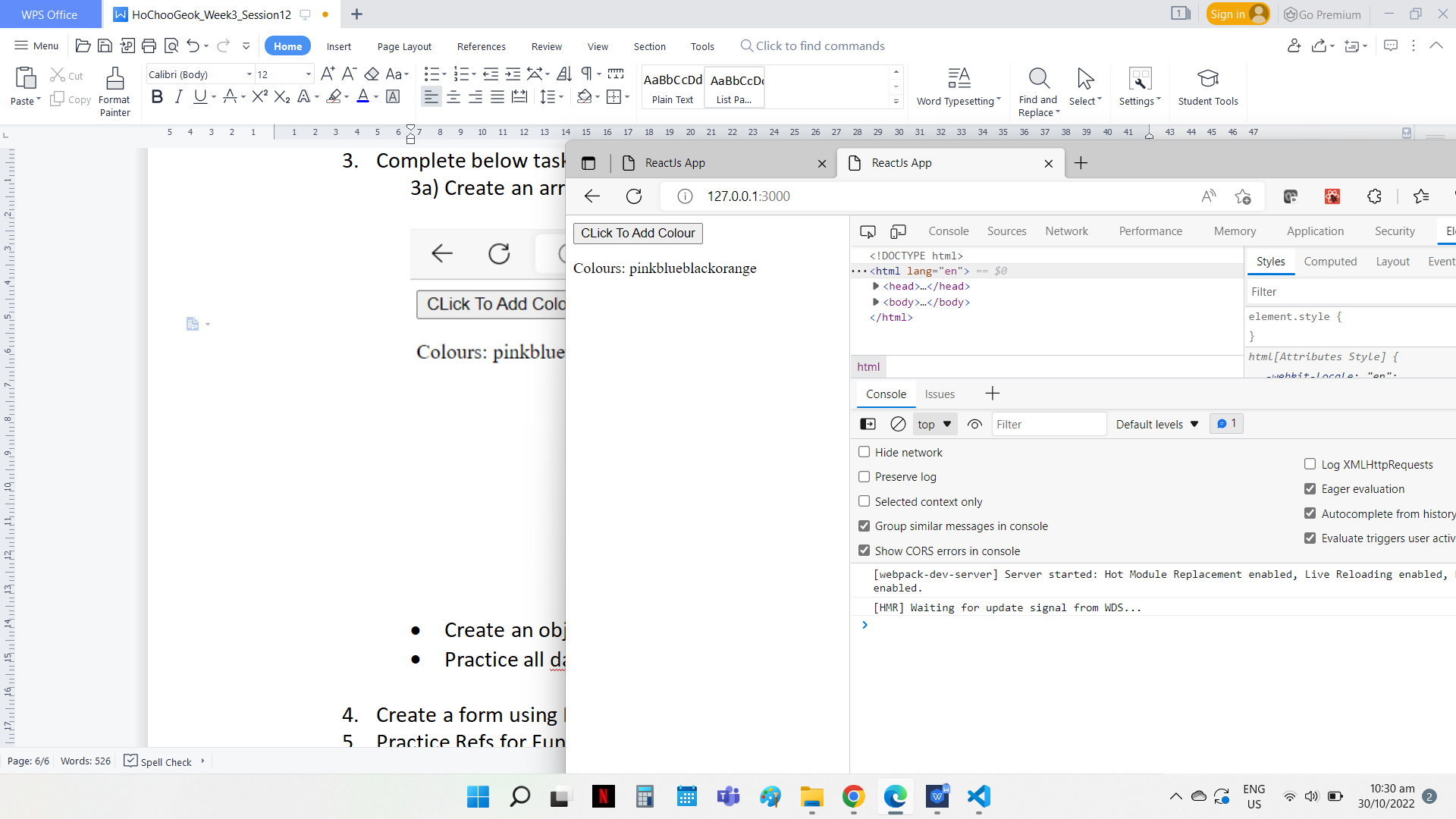
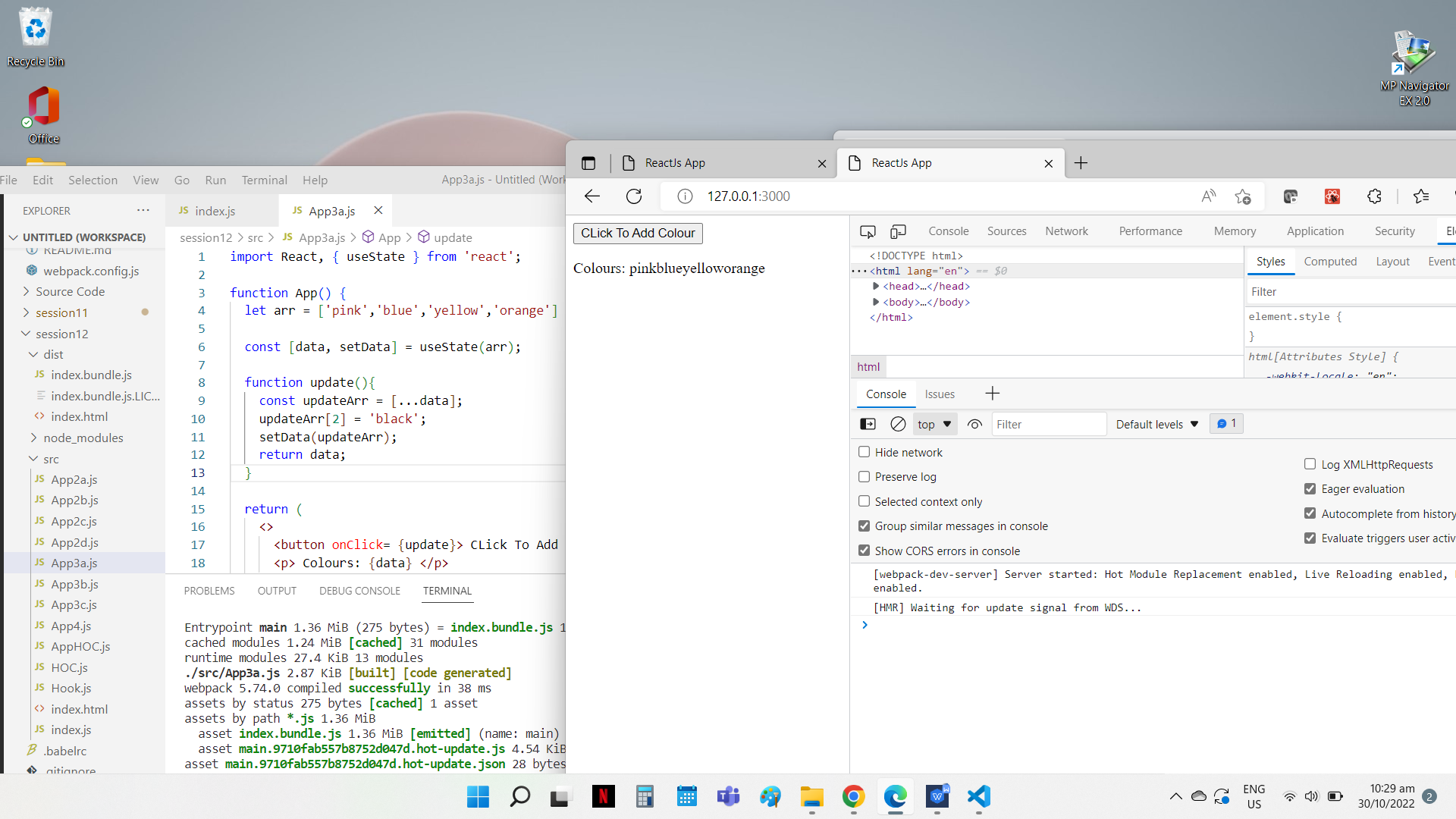
};

export default App;

2e) Create a search box (Optional)

3) Complete below tasks using Hooks

3a) Create an array of object and update its second index



Click button to change the 2nd index from yellow to black

Source Code

File Name: App3a.js

import React, { useState } from 'react';

function App() {

  let arr = ['pink','blue','yellow','orange']

  const [data, setData] = useState(arr);

  function update(){

    const updateArr = [...data];

    updateArr[2] = 'black';

    setData(updateArr);

    return data;

  }

  return (

    <>

      <button onClick= {update}> CLick To Add Colour </button>

      <p> Colours: {data} </p>

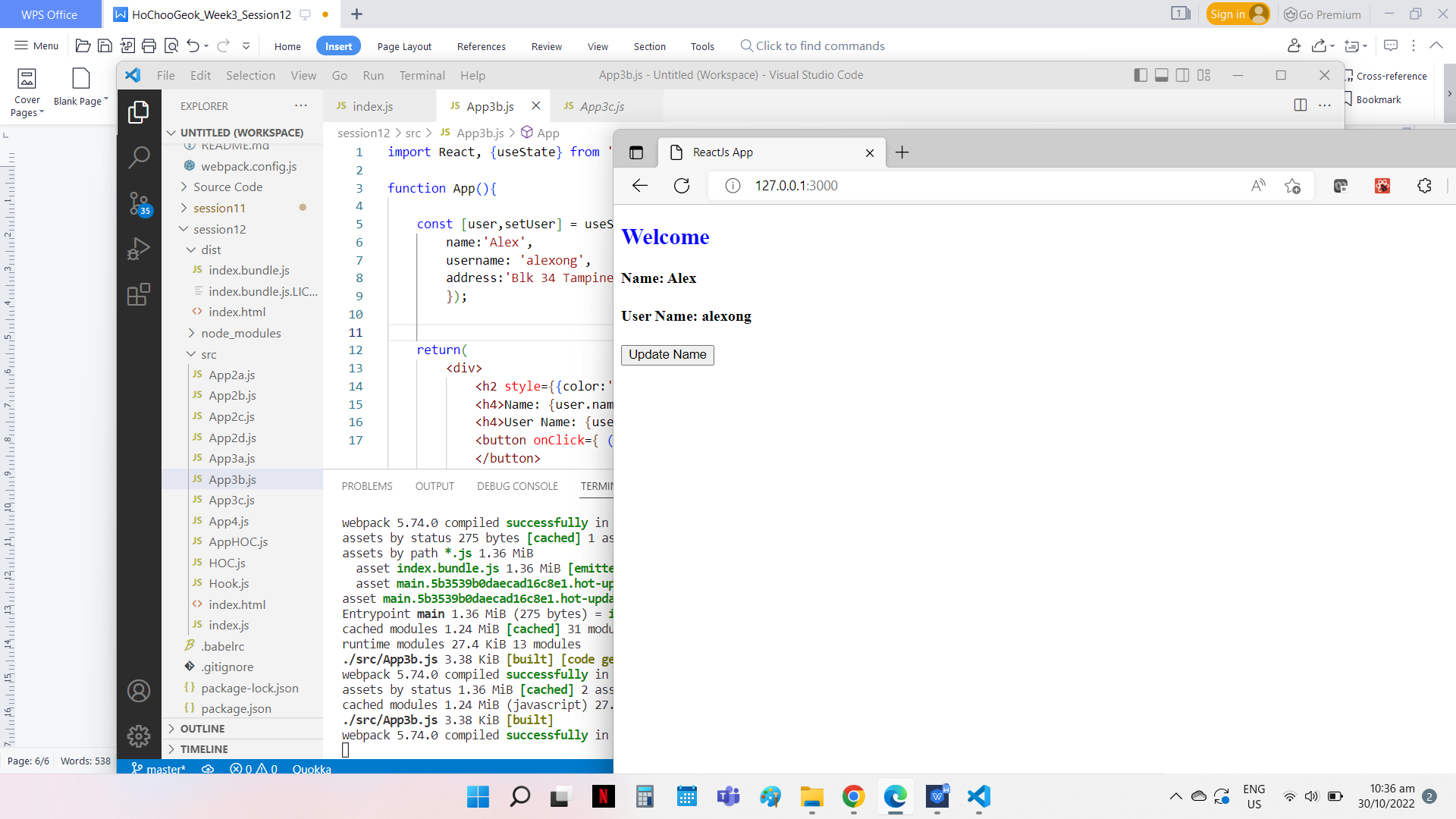
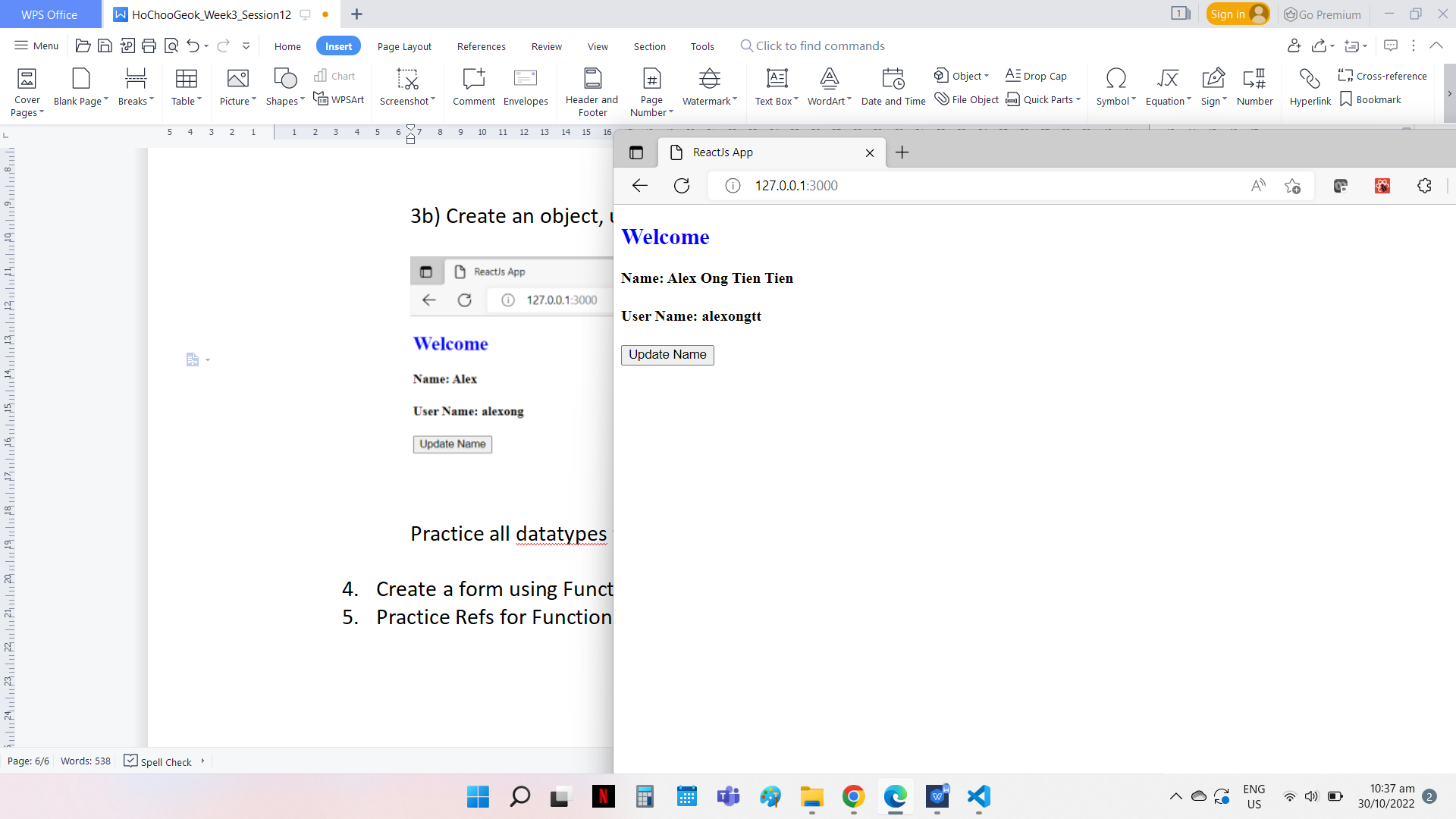
    </>

  );

};

export default App;

3b) Create an object, update it's 2 property values



Click button to update the name and username property values

Source Code

File Name: App3b.js

import React, {useState} from 'react';

function App(){

    const [user,setUser] = useState({

        name:'Alex',

        username: 'alexong',

        address:'Blk 34 Tampines'

        });

       return(

        <div>

            <h2 style={{color:'blue'}}> Welcome </h2>

            <h4>Name: {user.name} </h4>

            <h4>User Name: {user.username} </h4>

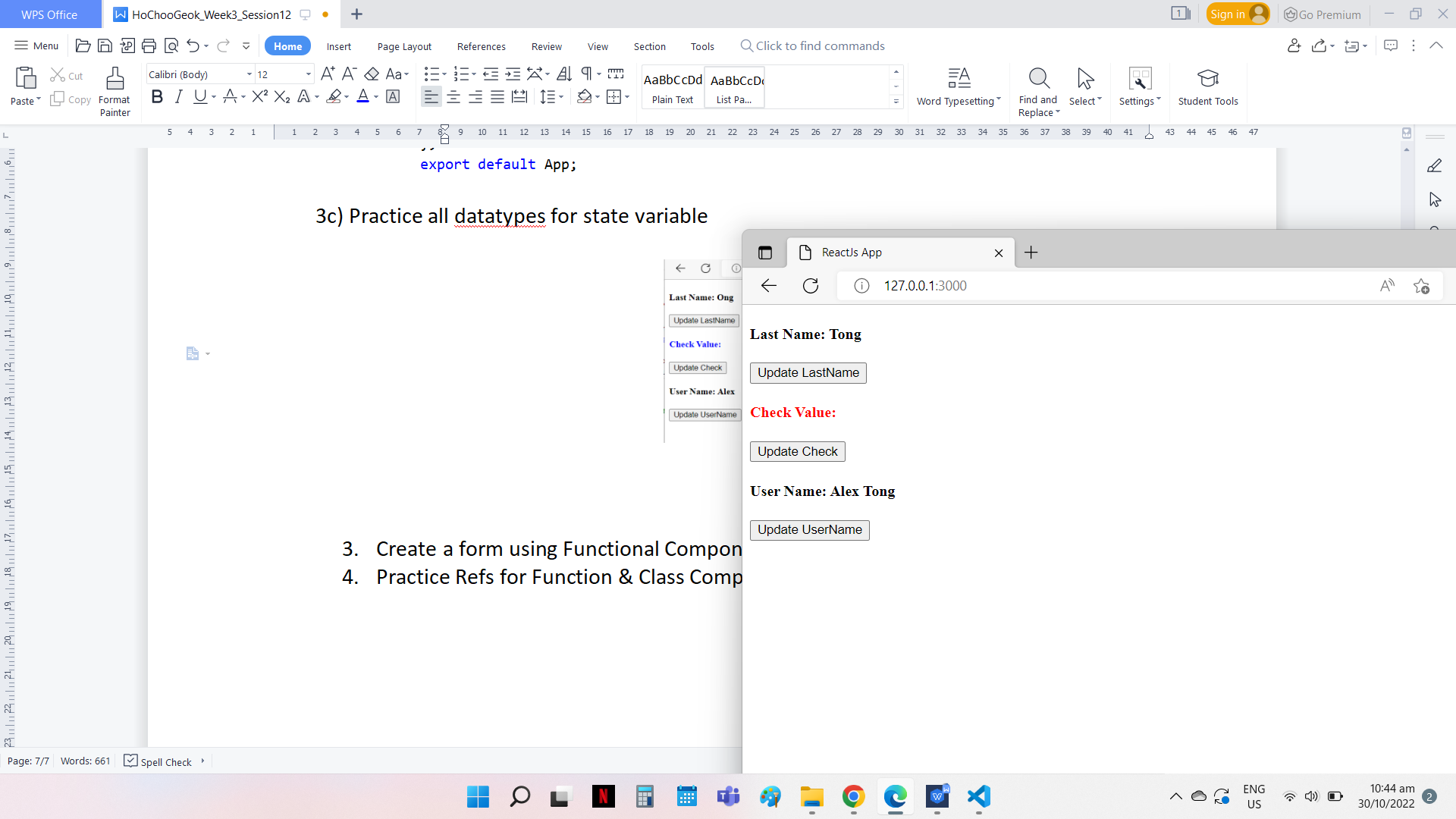
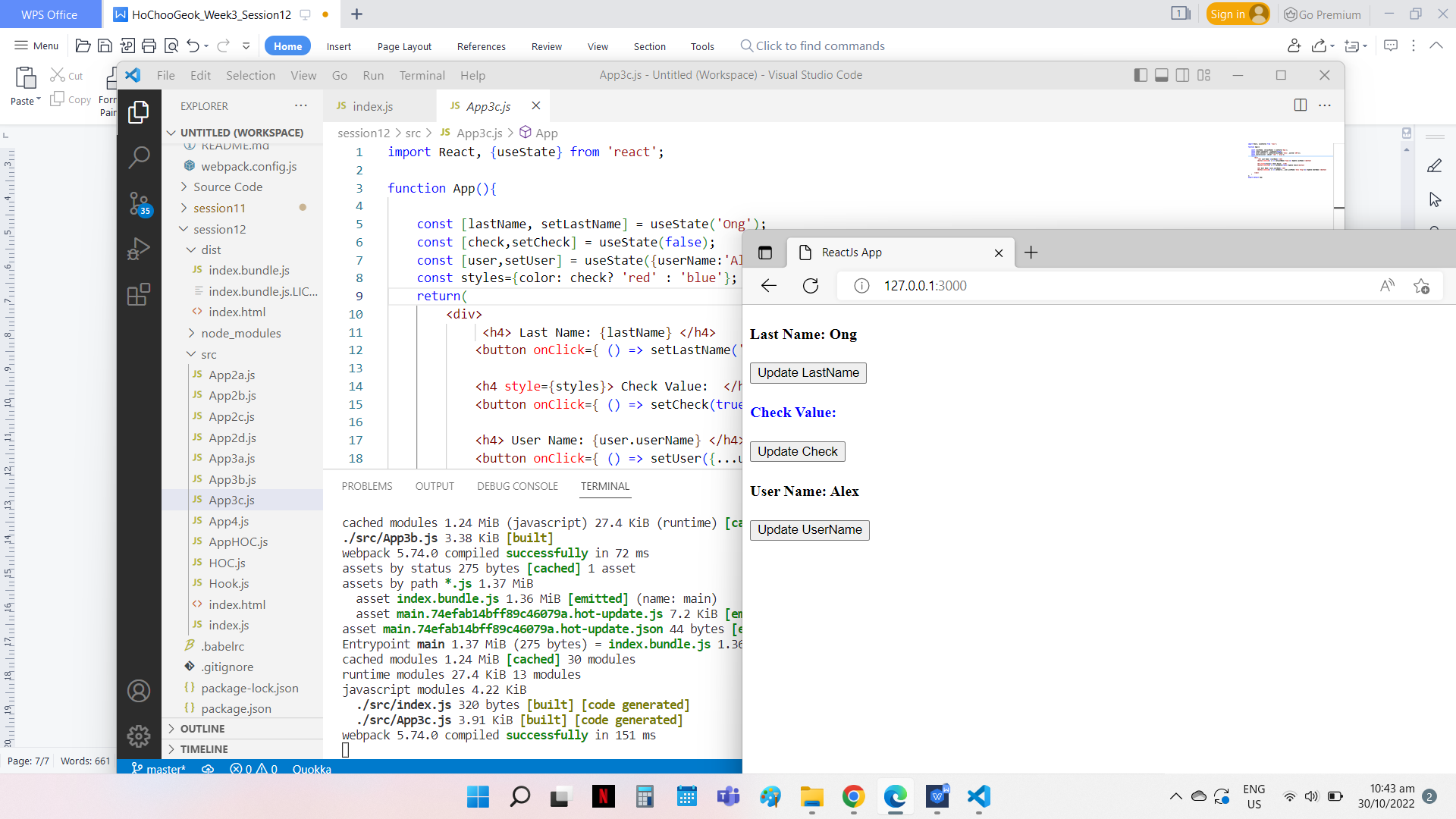
            <button onClick={ () => setUser({...user,name:'Alex Ong Tien Tien',username:'alexongtt'})}> Update Name </button>

        </div>

    )

};

export default App;

3c) Practice all datatypes for state variable

Source Code

File Name: App3c.js

import React, {useState} from 'react';

function App(){

    const [lastName, setLastName] = useState('Ong');

    const [check,setCheck] = useState(false);

    const [user,setUser] = useState({userName:'Alex', userId:'200'});

    const styles={color: check? 'red' : 'blue'};

    return(

        <div>

             <h4> Last Name: {lastName} </h4>

            <button onClick={ () => setLastName('Tong')}> Update LastName </button>

            <h4 style={styles}> Check Value:  </h4>

            <button onClick={ () => setCheck(true)}> Update Check</button>

            <h4> User Name: {user.userName} </h4>

            <button onClick={ () => setUser({...user,userName:'Alex Tong'})}> Update UserName </button>

</div>

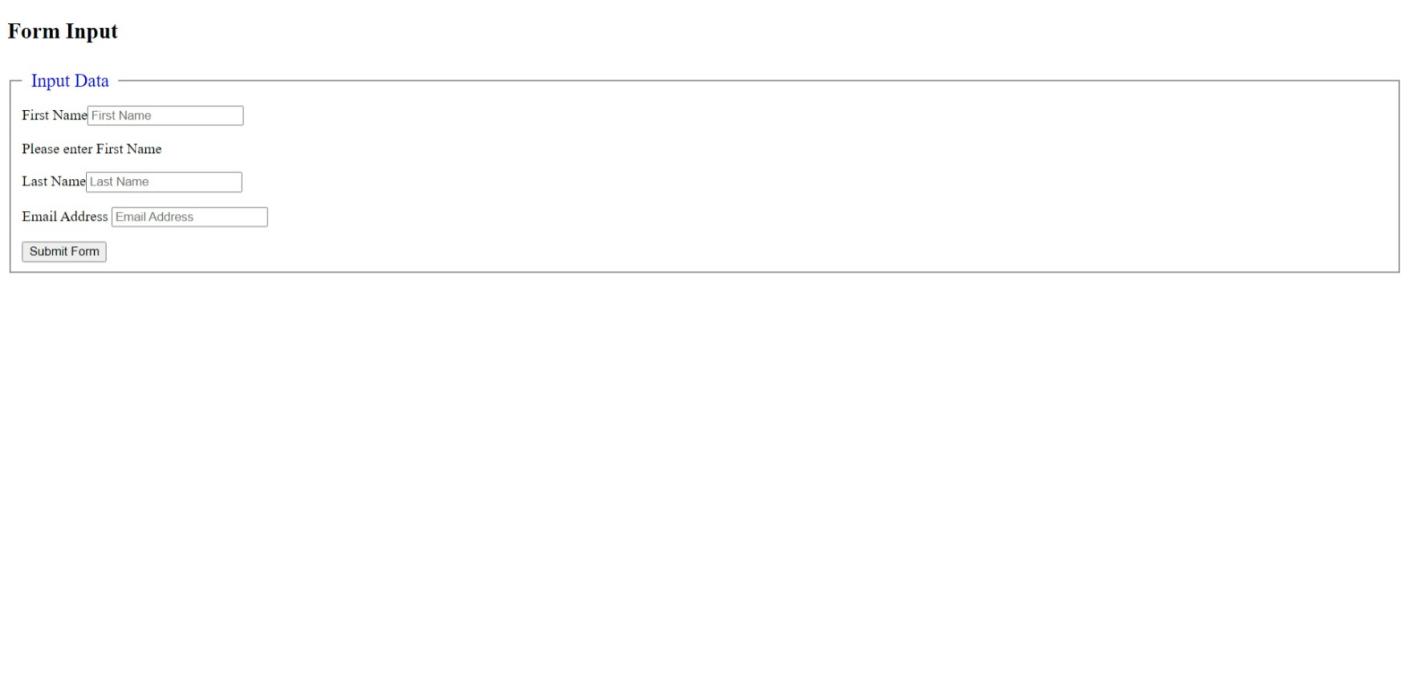
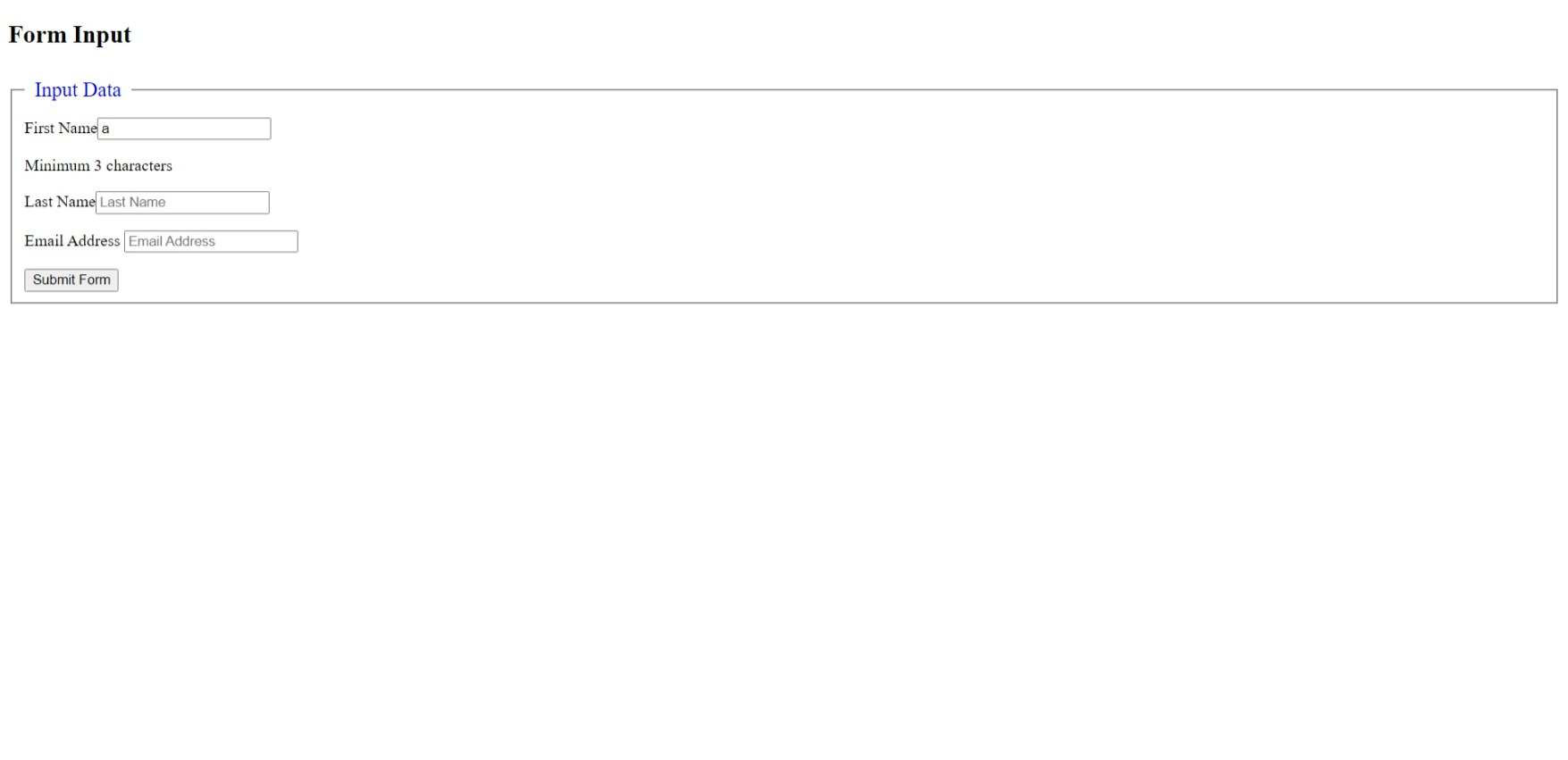
    )

};

export default App;

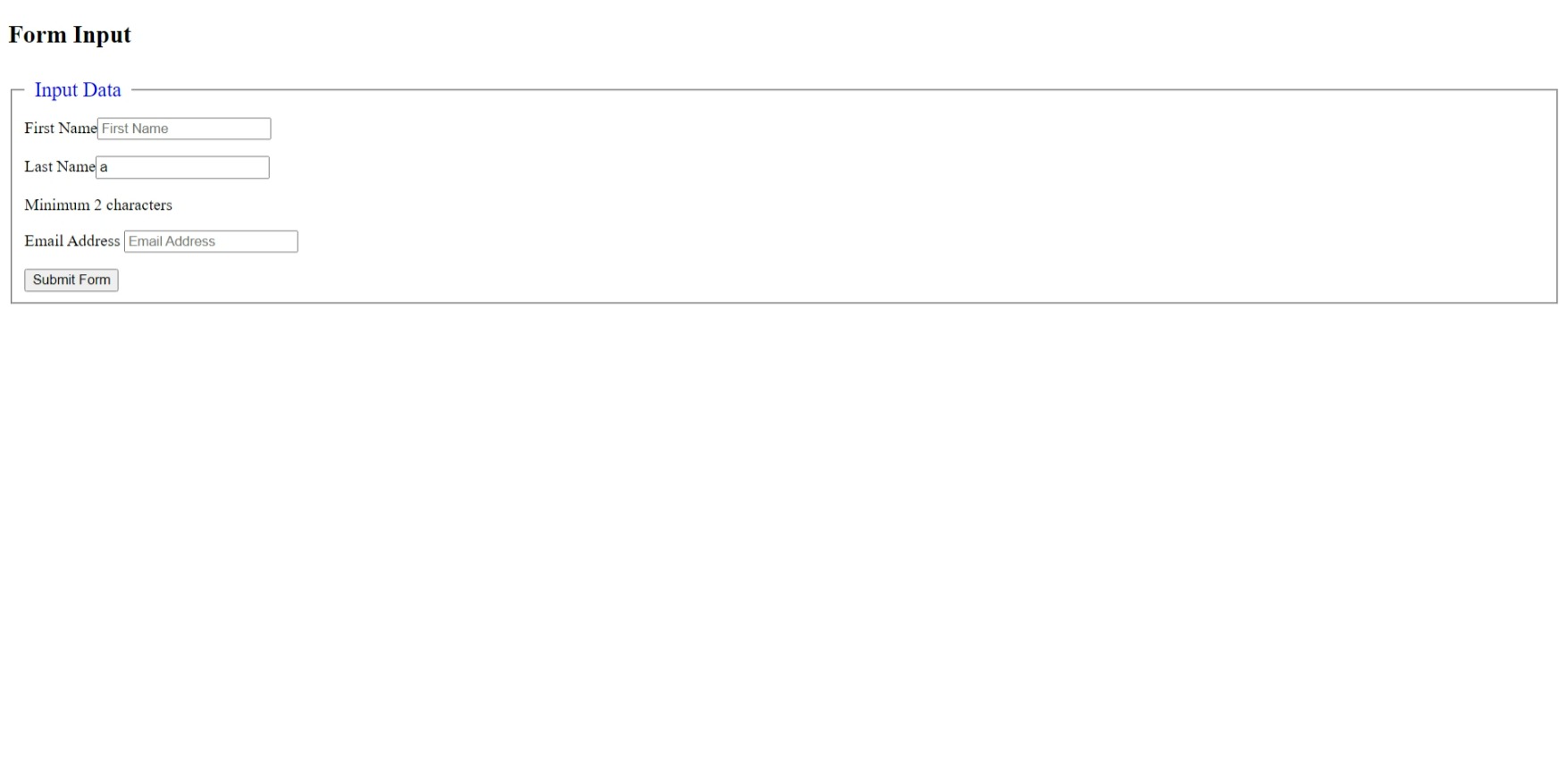
4. Create a form using Functional Component. Add validation. (Controlled Input)

A form using Functional Component was created with validations. Screen capture of the form as follow.

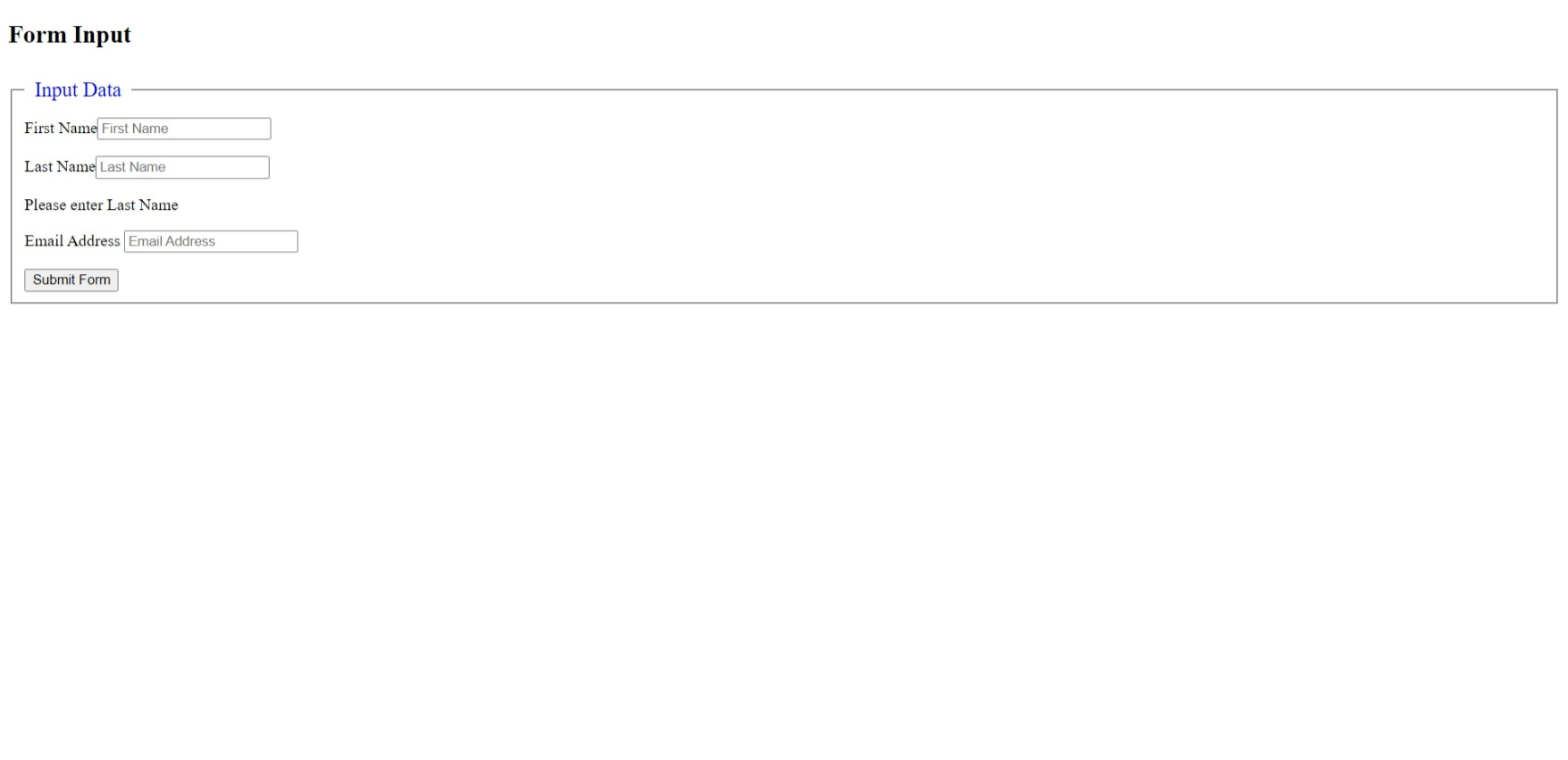


First Name Input validation for Minimum 3 characters

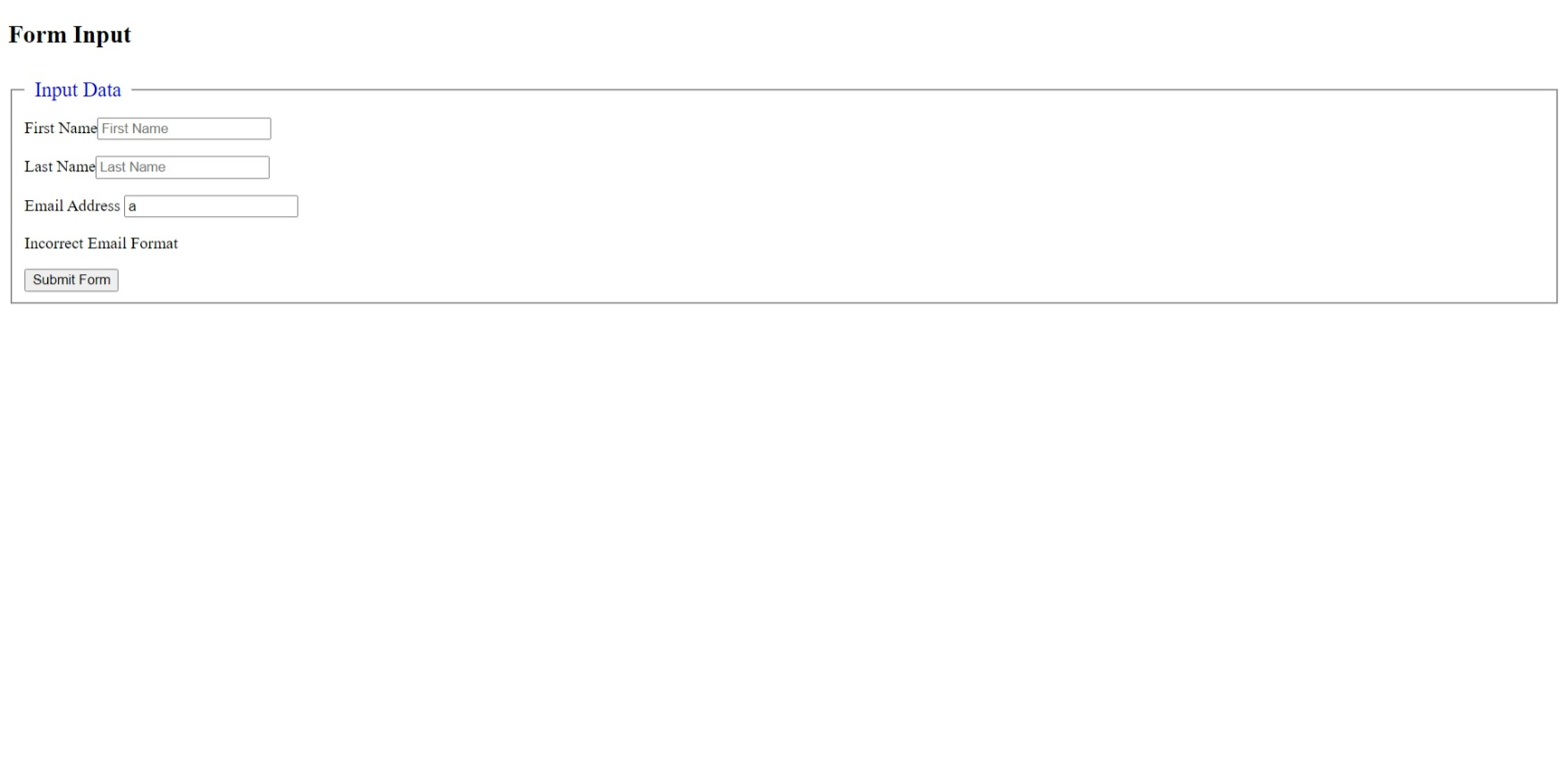
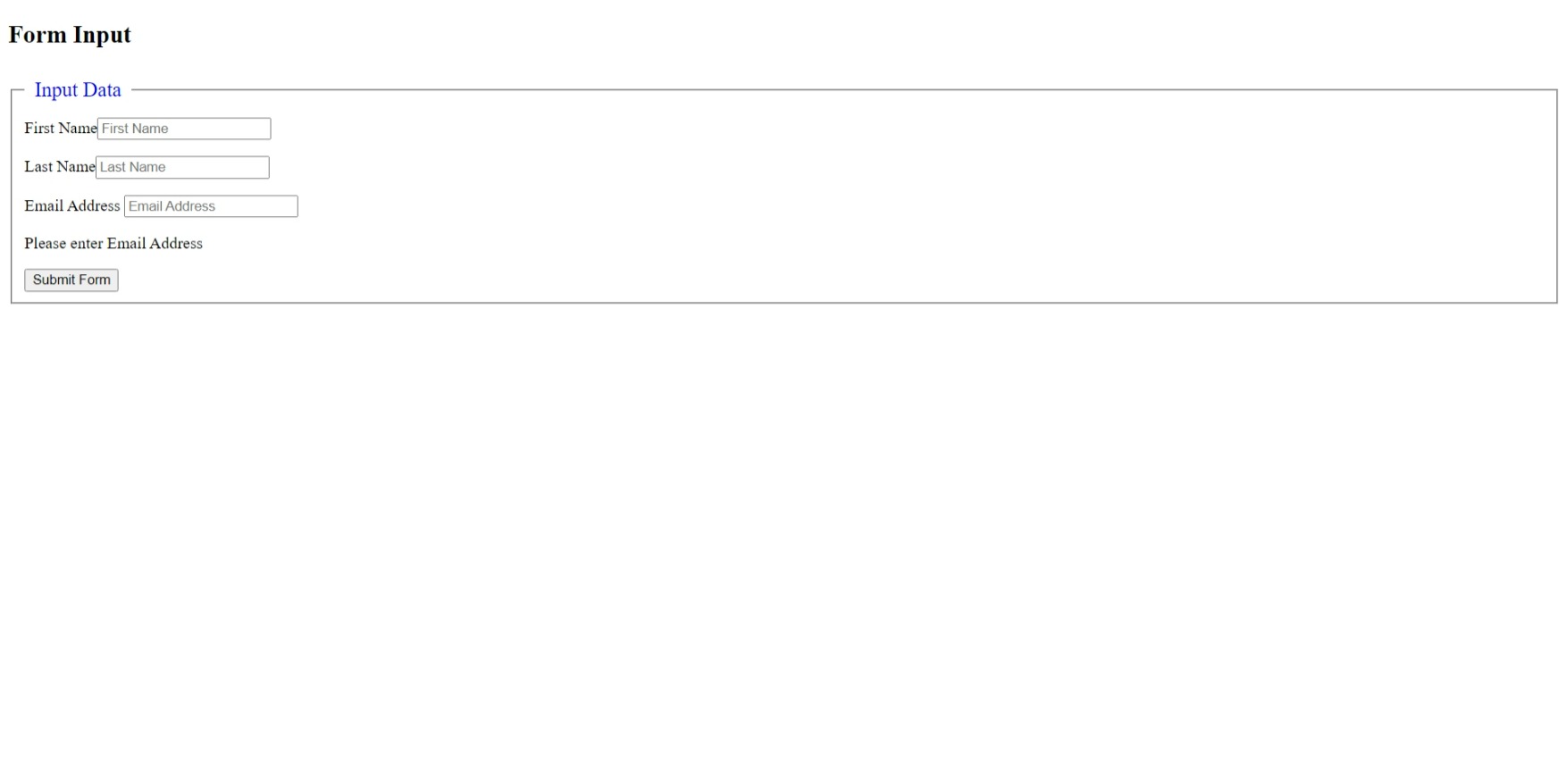
First Name Input validation for empty field



Last Name Input validation for Minimum 2 characters



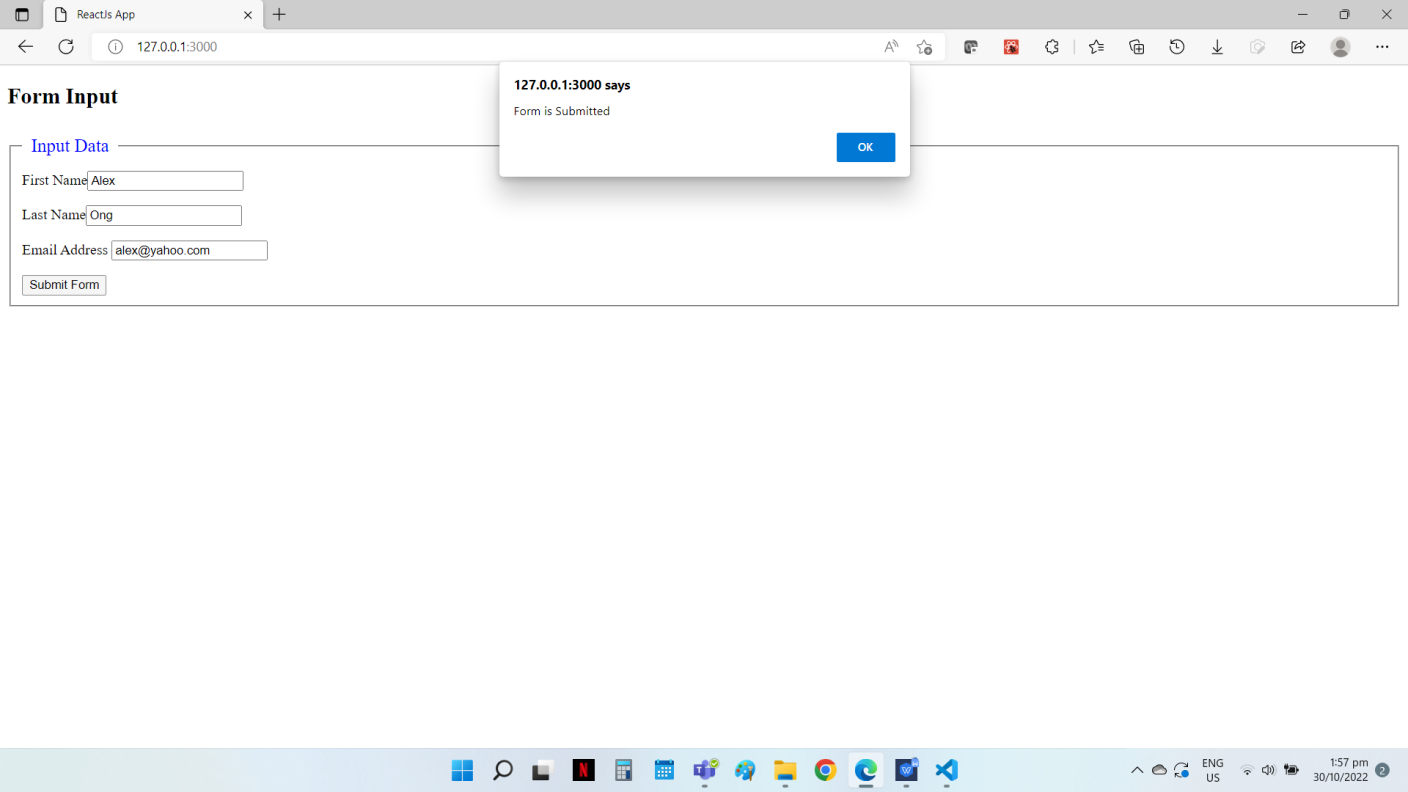
Last Name Input validation for empty field



Email input validation for incorrect format

Email input validation for empty field

Form Submitted Alert Box



Source Code

File Name: App4.js

import React, { useState } from "react";

function App() {

    const initialValue = { firstName: "", lastName: "" };

    const [formData, setFormData] = useState(initialValue);

    const [formError, setFormError] = useState({});

    const [formCheck, setFormCheck] = useState(false);

    // For user input check

    function handleChange(e) {

        const { name, value } = e.target;

        if ([name] == "firstname") {

            setFormError(checkfname(value));

        } else if ([name] == "lastname") {

            setFormError(checklname(value));

        } else if ([name] == "email") {

            setFormError(checkemail(value));

        }

    }

    //Validation check for First Name Field

    const checkfname = (updatefname) => {

        let errors = {};

        if (!updatefname) {

            errors.firstName = "Please enter First Name";

            setFormCheck(false);

        } else if (updatefname.length < 3) {

            errors.firstName = "Minimum 3 characters";

            setFormCheck(false);

        } else {

            errors.firstName = "";

            setFormCheck(true);

        }

        return errors;

    };

    // Validation check for Last Name field

    const checklname = (updatelname) => {

        let errors = {};

        if (!updatelname) {

            errors.lastName = "Please enter Last Name";

            setFormCheck(false);

        } else if (updatelname.length < 2) {

            errors.lastName = "Minimum 2 characters";

            setFormCheck(false);

        } else {

            errors.lastName = "";

            setFormCheck(true);

        }

        return errors;

    };

    // Validation check for Email field

    const checkemail = (updateemail) => {

        let regex =

            /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

        let errors = {};

        if (!updateemail) {

            errors.email = "Please enter Email Address";

            setFormCheck(false);

        } else if (!regex.test(updateemail)) {

            errors.email = "Incorrect Email Format";

            setFormCheck(false);

        } else {

            errors.email = "";

            setFormCheck(true);

        }

        return errors;

    };

    // To handle the submit button click

    function handleSubmit(e) {

        e.preventDefault();

        if (formCheck === true) {

            alert("Form is Submitted");

        } else {

            alert("Data incorrect");

        }

    }

    return (

        <div>

            <h2> Form Input </h2>

            <form onSubmit={handleSubmit}>

                <fieldset>

                    <legend style={{color:"blue", fontSize:20, padding:10}}> Input Data </legend>

                <label htmlFor="firstname"> First Name</label>

                <input

                    type="text"

                    name="firstname"

                    placeholder="First Name"

                    value={setFormData.firstName}

                    onChange={handleChange}

                />

                <p> {formError.firstName}</p>

                <label htmlFor="lastname"> Last Name</label>

                <input

                    type="text"

                    name="lastname"

                    placeholder="Last Name"

                    value={setFormData.lastName}

                    onChange={handleChange}

                />

                <p> {formError.lastName}</p>

                <label htmlFor="email"> Email Address </label>

                <input

                    type="email"

                    name="email"

                    placeholder="Email Address"

                    value={setFormData.email}

                    onChange={handleChange}

                />

                <p> {formError.email}</p>

                <button> Submit Form </button>

                </fieldset>

            </form>

        </div>

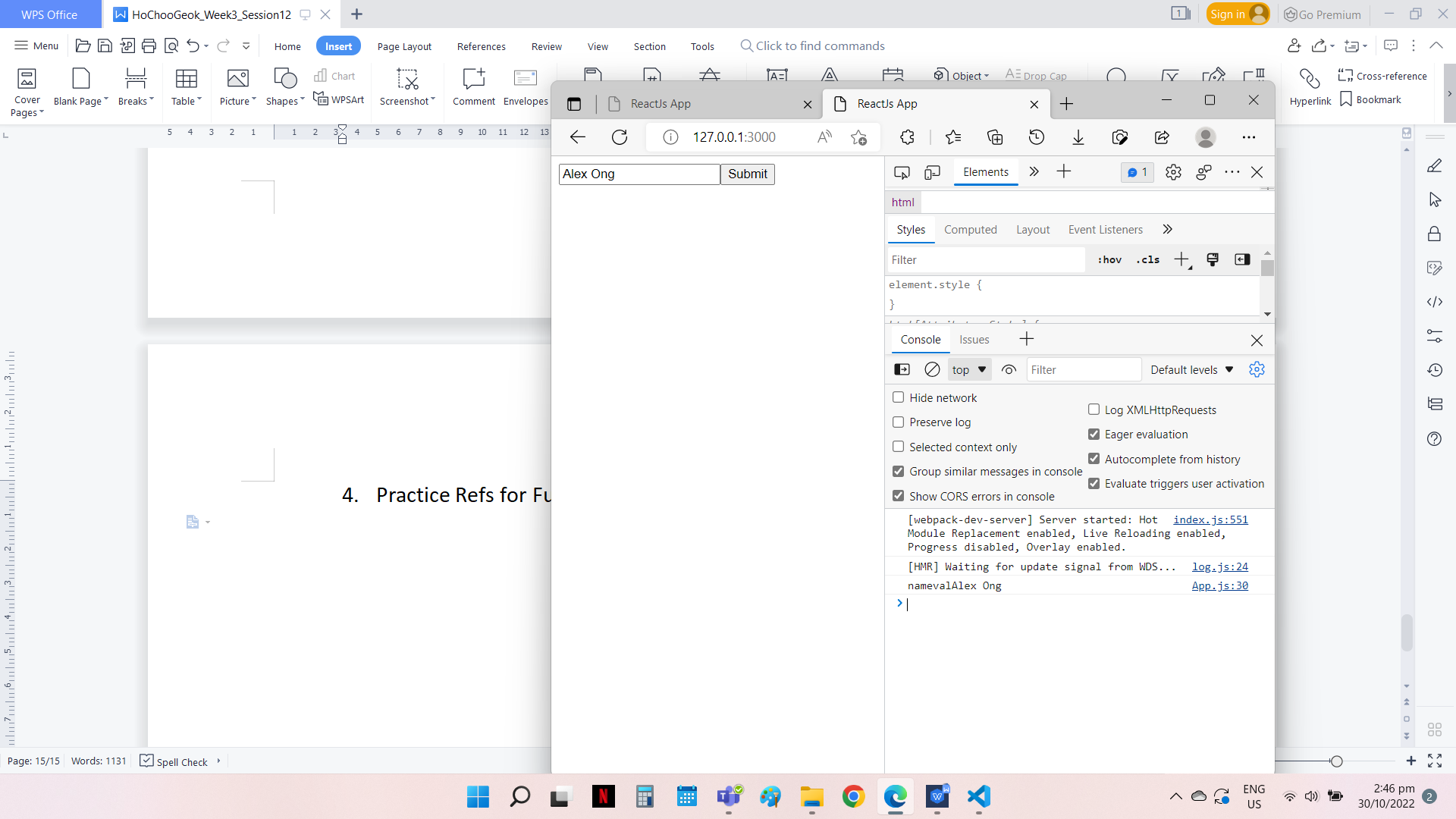
    );

}

export default App;

1. Practice Refs for Function & Class Component

5a) Use of Ref for Class based component

Source Code

File Name: App5a.js

import React, { Component , createRef} from 'react';

class App extends Component{

constructor(){

    super();

    this.name = createRef();

}

    handleClick = (e)=>{

        e.preventDefault();

        let nameVal = this.name.current.value

        console.log("nameval" + nameVal);

        this.name.current.focus();

    }

    render() {

        return (

             <div>

                <form>

                    <input type="text" ref={this.name} placeholder="Name"/>

                    <button onClick={this.handleClick}> Submit </button>

                </form>

             </div>

        );

    }

};

export default App;

5b) Use of Ref for Functional based component

Source Code

File Name: App.js

import React, {useRef} from 'react';

const App = () => {

    const name = useRef();

    function handleClick(e){

        name.current.focus();

        let nameVal = name.current.value;

        console.log(nameVal);

        }

    return(

        <div>

            <form>

            Name:  <input type="text" ref={name} />

            <button onClick={handleClick}> Submit</button>

            </form>

        </div>

    )

};

export default App;