Final Test for Course: Enterprise Application Development

Length: 90 minutes. Total marks: 40

Test rules:

1. This is an **open book and INDIVIDUAL**. Students can use Canvas and other offline documents to find their answers.
2. Students write their name and id and answer the questions in this document.
3. Exchanging answers or ideas between students, online chatting or posting questions to get support from other people, family, friends, or verbal chat during the test time is **STRICTLY prohibited**, and will be considered as cheating.
4. Before starting the test, **students will need to install Zoom or Teams on their computers**. In the 5 minutes before the test begins, students Start a Meeting in Zoom or Teams. They need to share their audio, video and their screens during the meeting. Press the Record **button** to record your meeting.

5. Make sure at the first part of the video, students move webcam around the room to show clearly enough to see front, rear, and two sides of the desks where students take the test. This is to make sure that no other people in the room support the student.
6. During the test, the screen sharing will ensure that students do not exchange information with other people.
7. **When the test is completed, students will submit the test answer first. Then when the video recording is ready, submit the video.**

8. For those who do not have a web camera, consider borrowing one or buy a new one.
9. Make sure you practice several times to get familiar with it

Test questions

Javascript (5+5 = 10 marks)

1. Write a JavaScript function to find an array contains a specific element

Test data :
arr = [2, 5, 9, 6];
console.log(contains(arr, 5));
[True]

……..……..
```
const contains = function (array, element){
  for(let i = 0; i < array.length; i++){
    if(array[i] === element){
      return true;
    }
  }
}
```
……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..
……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..
……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..

2. Write a JavaScript function to find the difference of two arrays

Test Data :
console.log(difference([1, 2, 3], [100, 2, 1, 10]));
["3", "10", "100"]

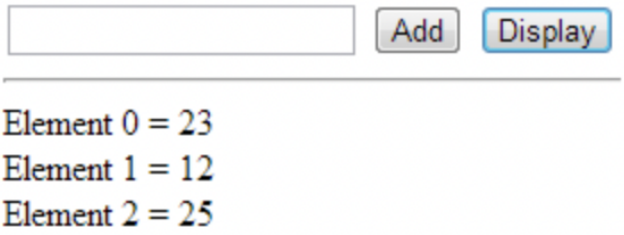
```
const difference = (array1, array2) => {
    let newArray = array1.filter((e) => {
      var isIn = false;
      for(let i = 0; i<array2.length; i++){
        if(e === array2[i]){
          isIn = true;
          return;
        }
      }
      if(!isIn){
        return e
      }
    })
    let newArray1 = array2.filter((e) => {
      var isIn = false;
      for(let i = 0; i<array1.length; i++){
        if(e === array1[i]){
          isIn = true;
          return;
        }
```

```
        }
        if(!isIn){
            return e
        }
    })
    for(let i = 0; i < newArray.length; i++){
        newArray1.push(newArray[i])
    }
    return newArray1
}
console.log(difference([1, 2, 3], [100, 2, 1, 10]));
```

……
……
……

ReactJS (5+10 = 15 marks)

3. Write a React components to add items in an blank array and display the items




```
import React, { useState } from 'react';
export default function Question3(){
    const [listOfItem, setListOfItem] = useState([])
    const [newItem, setNewItem] = useState()
    const [displayed, setDisplayed] = useState(false)
    const addItem = (value) => {
        setListOfItem(listOfItem => [...listOfItem, value])
    }
    const display = () =>{
        setDisplayed(true)
    }
    return(
        <div>
            <input type="text" onChange ={e => setNewItem(e.target.value)}></input>
            <button type="button" onClick = {() => addItem(newItem)}>Add</button>
            <button type="button" onClick = {() => display()}>Display</button>
            {displayed ? listOfItem.map((i, index) => <div>
                Element {index} = {i}
            </div>) : ""}
        </div>
    )
}
```

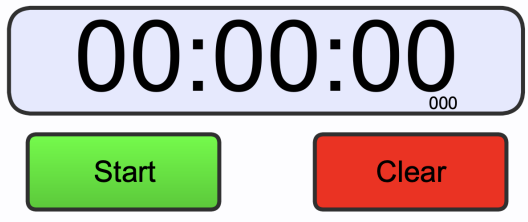
……
……
……

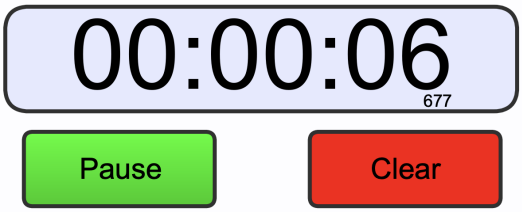
4. Write a React component to display a stopwatch.

When users click the **Start** button the **milliseconds number** will be increased by 1 every millisecond. When it reaches 1000, it is reset to 000 and the second number will be increased by 1.

Clear button will reset all numbers to 0 and stop running.

P/S You must use CSS to design the component.






```
import { useEffect, useState } from "react";

export default function Question4() {
    const [ms, setMs] = useState(0);
    const [started, setStarted] = useState(false)
    const countTime = () => {
        var timing;
        if (started) {
            timing = setTimeout(function () {
                setMs((ms + 1))
            }, 1);
        }
        else {
            clearTimeout(timing)
        }
    }
    useEffect(() => {
        countTime()
    }, [started, ms])
    const clear = () => {
        setMs(0);
        setStarted(false)
    }
    return (
        <div>
```

```
          <div style={{ width: 500, height: 20, marginLeft: "auto", marginRight: "auto", border: "1px black solid" }}>
            <div style={{ backgroundColor: "#e0ffff", fontSize: "50px" }}>{String(Math.floor((ms / (1000 * 60 * 60)) %
60)).padStart(2, '0')} : {String(Math.floor((ms / (1000 * 60)) % 60)).padStart(2, '0')} : {String(Math.floor((ms / 1000) %
60)).padStart(2, '0')}</div>
            <div style={{ backgroundColor: "#e0ffff", borderRight: "1px black solid", borderLeft: "1px black solid",
borderBottom: "1px black solid"}}>{String(ms % 1000).padStart(3, '0')} </div>
          </div>
          <div style={{ marginTop: "80px" }}>
            <button type="button" style={{ backgroundColor: "greenyellow", border: "1px, black solid", borderRadius:
"15%", padding: "5px 20px" }} onClick={started ? () => setStarted(false) : () => setStarted(true)}>{started ? "Pause" :
"Start"}</button>
            {/* <button type="button" onClick={() => setStarted(false)}>Pause</button> */}
            <button type="button" style={{ backgroundColor: "red", border: "1px, black solid", borderRadius: "15%",
padding: "5px 20px", marginLeft: "10px" }} onClick={() => clear()}>Clear</button>
          </div>
        </div>
    )
}
```

NodeJS-MongoDB (5+10 = 15 marks)

5. Write a middleware named **mylog()** to log all the incoming requests and save them to the MongoDB collection named **mylog**.
   Each document should include: date time, source IP address, URL. If a request has IP address in the list of black list, reject it. The black list is a collection of MongoDB named **blackedlist** that contains a list of banned IP addresses.
   Students must define the required 2 required schema.

```
……..……..…… const app = require('express')()
const bodyParser = require('body-parser')
app.use(bodyParser.json())
const mongoose = require('mongoose')

var MyLogSchema = new mongoose.Schema({
   dataTime : Date,
   sourceIPAddress: String,
   URL: String
})

 var BlackListSchema = new mongoose.Schema({
   bannedIPAddress: String
})

 var blacklist = mongoose.model('blackedlists', BlackListSchema)
 var mylogModel = mongoose.model('mylogs', MyLogSchema)


mongoose.connect("mongodb+srv://myuser:mypassword@cluster0.1lbnn.mongodb.net/sampletest").then(app.listen(
9000, () => function(){
    console.log('Server is running')
}))
 blacklist.insertMany({bannedIPAddress: "103.302.232.12"}, {bannedIPAddress:"103.301.231.23"})

const mylog = (req, res, next) => {
  blacklist.findOne({bannedIPAddress: req.body.sourceIPAddress}, function(error, found){
     if(error){
```

```
        return res.send(error)
      }
      if(found){
        return res.send('Rejected')
      } else {
        mylogModel.create(req.body, function(err, mylog){
         next()
       })
    }
  })
}
app.use(mylog)
app.post("/", function(req,res){
   res.send('Successfully Added')
})
```

……

……

……

6. Write a small REST API to return average, max, min ages of all users (in json format) in the MongoDB
   URL: http://localhost:4001/statistic?agg=max
   URL: http://localhost:4001/statistic?agg=min
   URL: http://localhost:4001/statistic?agg=avg

   User collection:
   [
   {
    name: "Toan",
    age: 30
   },
   {
    name: "Huy",
    age: 20
   }
   ]

```
const app = require('express')()
const bodyParser = require('body-parser')
app.use(bodyParser.json())
const mongoose = require('mongoose')

var UserSchema = new mongoose.Schema({
   name: String,
   age: Number
})
var User = mongoose.model('users', UserSchema)

mongoose.connect("mongodb+srv://myuser:mypassword@cluster0.1lbnn.mongodb.net/sampletest").then(app.listen(
4001, () => function(){
   console.log('Server is running')
```

```
}))
User.insertMany([{name: "Toan",age: 30},{name: "Huy", age: 20}])

app.get("/statistics", function(req, res){
    if(req.query.agg === "max"){
        User.find({}).sort({'age': -1}).exec(function(error, result){
            if(result){
                return res.send(result[0])
            }
        })
    }
    if(req.query.agg === "min"){
        User.find({}).sort({'age': 1}).exec(function(error, result){
            if(result){
                return res.send(result[0])
            }
        })
    }
    if(req.query.agg === "avg"){
        User.find({}, function(error, result){
            if(error){
                return res.send(error)
            }
            if(result){
                var totalAge = 0;
                for(let i = 0; i < result.length; i++){
                    totalAge = totalAge + result[i].age
                }
                var average = totalAge/result.length
                return res.send({average: average})
            }
        })
    }
})
```

. . . . . .

. . . . . .

. . . . . .

*****