ÔN TẬP CÔNG NGHỆ MỚI GIỮA KÌ – KHOA KỸ THUẬT PHẦN MỀM

**Kỹ Thuật** : Sử dụng Node js ( sử dụng thư viện ExpressJs ,Multer,AWS dotenv,path) DynomoDB ( Database ), S3 ( Lưu hình ảnh vào S3 ) , IAM user ( lấy SECRET\_ACCESS\_KEY và ACCESS\_KEY\_ID kết nối thông qua thư viện aws-sdk)

**Nội dung :** Tạo 1 form html nhập dữ liệu và hiện dữ liệu ra table ở dưới

A screenshot of a computer

Description automatically generated

**1 Số lưu ý khi ae config aws**

+ ) Đầu tiên mọi người phải phải đảm bảo thường xuyên kiểm tra tài khoản AWS của mình hoạt động ổn định tránh trường hợp gần vào thi thì tài khoản bị tạm ngưng ( t đã bị điều này khi đang làm bài tập và gửi email lên đội CSKH của AWS và đợi khá lâu mới nhận được phản hồi cỡ 3-5 ngày tuy nhiên tới hiện tại gần 1 tuần vẫn chưa xử lý xong ) , Nếu như không may tài khoản AWS bị tạm ngưng có thể mượn bạn bè tạo 1 tài khoản IAM USER cấp đủ 2 quyền truy cập là AmazonS3FullAccess và AmazonDynamoDBFullAccess và khi bạn B cấp cho bạn A tài khoản IAM USER thì nhớ cung cấp luôn cả 2 key quan trọng là Access key **và** Secret access key . Trường hợp bất khả kháng đó là ae đang làm mà mất mạng không kết nối được lên AWS thì vấn đề này thầy Phước đã có nhắc trong tiết thực hành cuối trước khi thi Giữa Kỳ là ae sẻ làm 1 data trên mảng thầy sẽ chấm điểm cho ae điểm thao tác với nodejs nhé . À còn điều nữa thì Thầy Phước cho phép AE sử dụng lại Access key và Secret access key lại nhé nên khuyến khích ae tạo trước ở nhà khi đi thi thì sử dụng lại 2 key này sẽ tiếp kiệm thời gian hơn

+ ) Điều thứ 2 mọi người cần lưu ý là khi tạo S3 bucket trong AWS thì mọi người phải để ý quyền truy cập bucket là **public** , điều này khi tạo 1 cái bucket mn phải bỏ tích mục **Block *all* public access** vàđiều chỉnh lại file policy ở dưới lưu ý đổi cái image1111 thành tên bucket của ae **(file tý t gửi ae sau)**

À ae nhớ chọn Object Ownership là ACLs enabled khi khởi tạo bucket nhé

A screenshot of a computer program

Description automatically generated

+ ) Còn về vấn đề tạo bảng thực hành với **DynamonDB** trên AWS thì này là khá là dễ. Nhắc sơ 1 chút trọng tâm ae chỉ cần lưu ý 1 table trong dynamonDb có 1 Partition key và 1 Sort key . AEchỉ cần quan tâm Partition key thôi nó giống như là 1 khoá chính như trong các cơ sở dữ liệu khác vậy và đương nhiên khoá chính là duy nhất   
Ví dụ như 1 Sinh Viên A chỉ có 1 MSSV duy nhất của chính mình , không thể có Sinh Viên B trùng MSSV với SV A được . Trong trường hợp này MSSV là Partition key trong bảng SinhVien

**OK giờ mình tiến hành tạo AWS**

**Code**

Đề bài mẫu : Tạo 1 form thông tin nhân sự bao gồm các thông tin sau

**nhansu(manhansu, hoten, phongban,namsinh,hinhanh)**

**trong đó phòng ban là 1 select box gồm : Phòng ban A, B,C**

**A screenshot of a computer

Description automatically generated**

AE nhớ check máy có nodeJs chưa nhé chưa có thì lên google cài

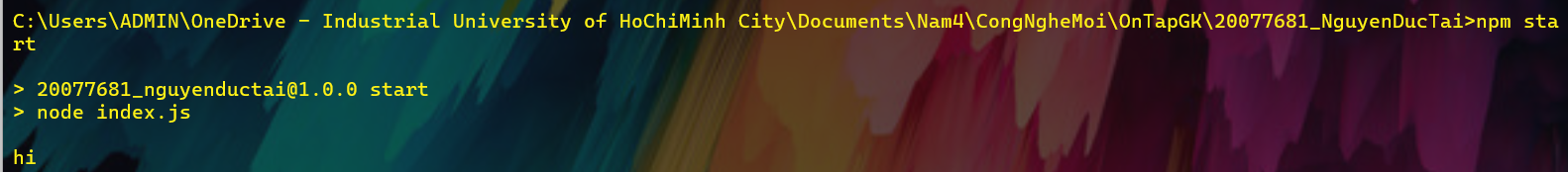
Tạo 1 thư mục xong ae vô cmd thực hiện như sau

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated



Sau khi ae thực hiện câu lệnh npm init và đã có file **package.json** rồi ae config xong rồi ae cài đặt các **packages** cho ứng dụng của mình

Trong terminal khi nãy ae nhập : **npm i ejs nodemon express** và trong mục start khi nãy ae đổi thành "start": "nodemon index" để nó tự chạy lại khi ae sửa file index.js

**Cấu Hình Server**

Như đã nói ở đầu bài thì mình sử dụng express JS nên ae config lại như sau

AE tạo 1 folder là **“views”** sau đó tạo file index.ejs và style.css và config lại file index.js như sau

**A screenshot of a computer screen

Description automatically generated**

Sau đó chạy ctrl + s lại bài ra được trang index.ejs sau

**A screenshot of a computer

Description automatically generated**

OK bây giờ mình tiến hành tạo giao diện, view thì đơn giản thôi ae tạo 2 form một form để nhập và 1 form bảng để hiện kết quả và 2 nút thêm vào xoá , mình cũng thêm 1 data mẫu vào trước nhé:

<!DOCTYPE *html*>

<html *lang*="en">

  <head>

    <meta *charset*="UTF-8" />

    <meta

*name*="viewport"

*content*="width=

    , initial-scale=1.0"

    />

    <link *rel*="stylesheet" *href*="style.css" />

    <title>Document</title>

  </head>

  <body>

    <h3>Nhân sự</h3>

    <form *action*="/save" *method*="post" *enctype*="multipart/form-data">

      <label> Nhập mã nhân sự : </label>

      <input

*type*="number"

*name*="manhansu"

*id*="manhansu"

*placeholder*=" Nhập mã nhân sự"

      />

      <br />

      <br />

      <label> Nhập họ tên : </label>

      <input *type*="text" *name*="hoten" *id*="hoten" *placeholder*=" Nhập họ tên" />

      <br />

      <br />

      <label> Nhập năm sinh : </label>

      <input

*type*="number"

*name*="namsinh"

*id*="namsinh"

*placeholder*=" Nhập  năm sinh"

      />

      <br />

      <br />

<label> Nhập phòng ban : </label>

      <select>

        <option *value*="">Phòng Ban A</option>

        <option *value*="">Phòng Ban B</option>

        <option *value*="">Phòng Ban C</option>

      </select>

      <label> Nhập hình ảnh : </label>

      <input *type*="file" *name*="hinhanh" *id*="hinhanh" />

      <br />

      <br />

      <input *type*="submit" *value*="Thêm" />

    </form>

    <br />

    <br />

    <form *action*="/delete" *method*="post" *enctype*="multipart/form-data">

      <table *border*="1">

        <thead>

          <tr>

            <th>Mã nhân sư</th>

            <th>Họ tên</th>

            <th>Năm sinh</th>

  <th>Phòng Ban</th>

            <th>Hình ảnh</th>

            <th>Chọn</th>

          </tr>

        </thead>

        <tbody>

          <tr>

            <td>1</td>

            <td>Nguyễn Vân Anh</td>

            <td>2004</td>

            <td>

              <img

*src*="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTXVLhVCMReryyxtn-uXeIxak9X\_n1ODrtf7w&usqp=CAU"

*alt*=""

*width*="100"

              />

            </td>

            <td>

              <input *type*="checkbox" *name*="manhansu" *value*="1" />

            </td>

          </tr>

        </tbody>

      </table>

      <br />

      <br />

      <input *type*="submit" *value*="Xoá" />

    </form>

  </body>

</html>

Sau khi chạy đoạn code trên thì ta có được 1 trang web đơn giản sau đây

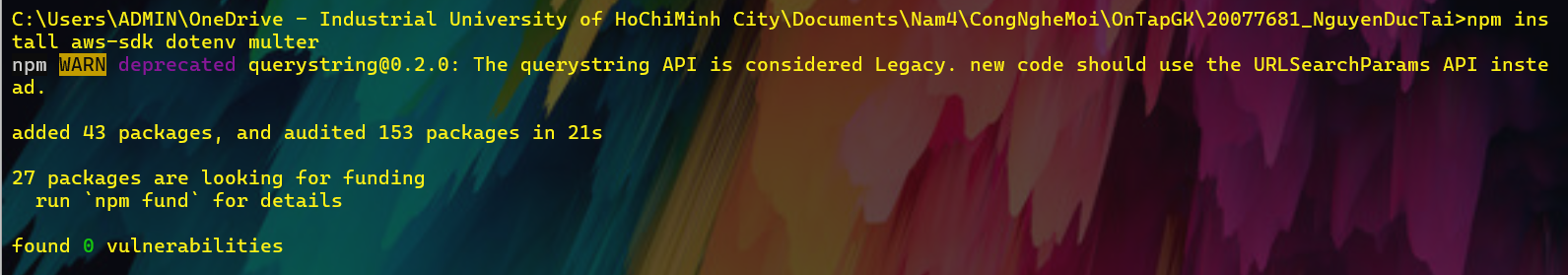
**A screenshot of a computer

Description automatically generated**

Ok bây giờ ta tiến hành code phần bên trong **server (chi tiết ae có thể xem ở file lab5 thầy Phước có gửi)**

Đầu tiên ae tiến hành add thêm 3 thư viện là aws-sdk và dotenv và multer để chuẩn bị kết nối lên Cloud AWS:

**npm install aws-sdk dotenv multer**

****

Sau đó ae sẽ tạo file .env trong root folder để lưu trữ các giá trị AccessKey, SecretKey tài khoản IAM User

A screenshot of a computer

Description automatically generated

Ae lưu ý như ở đầu bài có nhắc tình huống nếu ae bị khoá acc , còn account ae ổn định thì như đã nói ở đầu video ae sẽ lấy 2 key này ở trong iamuser lúc khởi tạo ví dụ ở đây t vừa khởi tạo 1 user3

A screenshot of a computer

Description automatically generated

- Chú ý region ap-southeast-1 là khu vực của Singapore

Chỉnh sửa index.js, khai báo thư viện aws, dotenv, multer và cấu hình AWS kết nối tới Cloud thông qua accessKey và secretAccessKey

const multer = require('multer')

const path = require('path')

const AWS = require('aws-sdk')

require('dotenv').config()

// cấu hình aws sdk

AWS.config.update({

  region: *process*.*env*.REGION,

  accessKeyId: *process*.*env*.ACCESS\_KEY\_ID,

  secretAccessKey: *process*.*env*.SECRET\_ACCESS\_KEY,

})

const s3 = **new** AWS.S3() // Khai báo s3

const dynamodb = **new** AWS.DynamoDB.DocumentClient() // khai báo Service dynamodb

const bucketName = *process*.*env*.S3\_BUCKET\_NAME

const tableName = *process*.*env*.DYNAMODB\_TABLE\_NAME

sau đó ae cấu hình multer để quản lý việc upload hình ảnh:

// cấu hình multer

const storage = multer.memoryStorage({

  destination: function (*req*, *file*, *callback*) {

    callback(null, '')

  },

})

const upload = multer({

  storage: storage,

  limits: { fileSize: 2000000 }, // giới hạn file 2MB

  fileFilter: function (*req*, *file*, cb) {

    checkFileType(*file*, cb)

  },

})

// hàm này có chức năng sẽ validate định danh file updalod có phải là ảnh không

function checkFileType(*file*, *cb*) {

  const filetypes = /jpeg|jpg|png|gif/ // kiểm tra file có phải là ảnh không

  const extname = filetypes.test(path.extname(*file*.originalname).toLowerCase())

  const mimetype = filetypes.test(*file*.mimetype)

  if (mimetype && extname) {

    return cb(null, true)

  }

  return cb('Error: Images Only!')

}

Sau đó ae render data lên trang index.ejs

app.get('/', async (*req*, *res*) => {

  try {

    const params = { TableName: tableName }

    const data = await dynamodb.scan(params).promise() // dùng hàm scan để lấy dữ liệu từ bảng

    console.log('data =', data.Items)

    return *res*.render('index.ejs', { data: data.Items })

  } catch (error) {

    console.log(error)

    return *res*.status(500).json({ message: 'Internal Server Error' })

  }

})

Console.log đã ra :   
A screen shot of a computer

Description automatically generated

ở trong table thì ae sửa lại 1 xíu ở tbody

  <% data.forEach(item => { %>

              <tr>

                <td>

                  <%=item.manhansu %>

                </td>

                <td>

                  <%=item.hoten %>

                </td>

                <td>

                  <%=item.namsinh %>

                </td>

                <td>

                  <%=item.phongban %>

                </td>

                <td>

                  <img *src*="<%=item.hinhanh %>" *alt*="" *width*="100" />

                </td>

                <td>

                  <input *type*="checkbox" *name*="manhansu" *value*="1" />

                </td>

              </tr>

              <% })%>

          </tr>

**Kết quả chạy ra như sau :**  
A screenshot of a computer

Description automatically generated

Ok giờ đã load được dữ liệu ta sẽ tiến hành thêm , xoá nhân sự ta chỉnh sửa ở file index.js

// upload.single('image') là middleware để upload 1 file duy nhất

app.post('/save', upload.single('image'), async (*req*, *res*) => {

  try {

    // lay du lieu tu form

    const manhansu = *Number*(*req*.body.manhansu)

    const hoten = *req*.body.hoten

    const namsinh = *Number*(*req*.body.namsinh)

    const phongban = *req*.body.phongban

    const image = *req*.file?.originalname.split('.') // kiểm tra file có tồn tại hay không sau đó nó chia chuỗi thành mảng dựa vào dấu chấm ví dụ ductai.jpg => ['ductai', 'jpg']

    const fileType = image[image.length - 1] // lấy đuôi file ví dụ: jpg, png, jpeg

    const filePath = `${manhansu}\_${*Date*.now().toString()}\_${fileType}` // đặt tên cho hình ảnh theo cấu trúc mã nhân sự + thời gian + đuôi file ví dụ: 6\_1234567890\_jpg

    //  cấu hình param cho S3

    const paramS3 = {

      Bucket: bucketName,

      Key: filePath,

      Body: *req*.file.buffer,

      ContentType: *req*.file.mimetype,

    }

    s3.upload(paramS3, async (*err*, *data*) => {

      // upload ảnh lên s3

      if (*err*) {

        console.error('Error uploading image to S3', *err*)

        return *res*.status(500).json({ message: 'Internal Server Error' })

      } else {

        const imageURL = *data*.Location // gán URL S3 vào đường dẫn image

        const params = {

          TableName: tableName,

          Item: {

            manhansu: manhansu,

            hoten: hoten,

            namsinh: namsinh,

            phongban: phongban,

            hinhanh: imageURL,

          },

        }

        await dynamodb.put(params).promise() // lưu dữ liệu vào bảng

        return *res*.redirect('/') //redirect về trang chủ

      }

    })

  } catch (error) {

    console.error('Error saving data from DynamoDb ', error)

    return *res*.status(500).json({ message: 'Internal Server Error' })

  }

})

Code bắt sự kiện xoá

// bắt sự kiện xoá

// upload.fields([]) là middleware để không upload file

app.post('/delete', upload.fields([]), (*req*, *res*) => {

  const listCheckboxSelected = *Object*.keys(*req*.body) // Lấy ra tất cả checkboxes

  //kiểm tra có chọn checkbox nào không , nếu không thì trả về trang chủ

  if (listCheckboxSelected.length <= 0 || !listCheckboxSelected) {

    return *res*.redirect('/')

  }

  try {

    // định nghĩa hàm đệ quy xoá dữ liệu

    function onDeleteItem(*length*) {

      const params = {

        TableName: tableName,

        Key: {

          manhansu: *Number*(listCheckboxSelected[*length*]),

        },

      }

      dynamodb.delete(params, (*err*, *data*) => {

        if (*err*) {

          console.error('Error deleting data from DynamoDb ', *err*)

          return *res*.status(500).json({ message: 'Internal Server Error' })

        } else {

          if (*length* > 0) {

            onDeleteItem(*length* - 1)

          } else {

            return *res*.redirect('/')

          }

        }

      })

    }

    // Nếu xoá thành công, hàm sẽ gọi lại chính nó với length giảm đi 1, tiếp tục xoá các mục khác. Hàm sẽ dừng khi length không còn lớn hơn 0.

    onDeleteItem(listCheckboxSelected.length - 1)

  } catch (error) {

    console.error('Error deleting data from DynamoDb ', error)

    return *res*.status(500).json({ message: 'Internal Server Error' })

  }

**Chúc các bạn thi tốt**