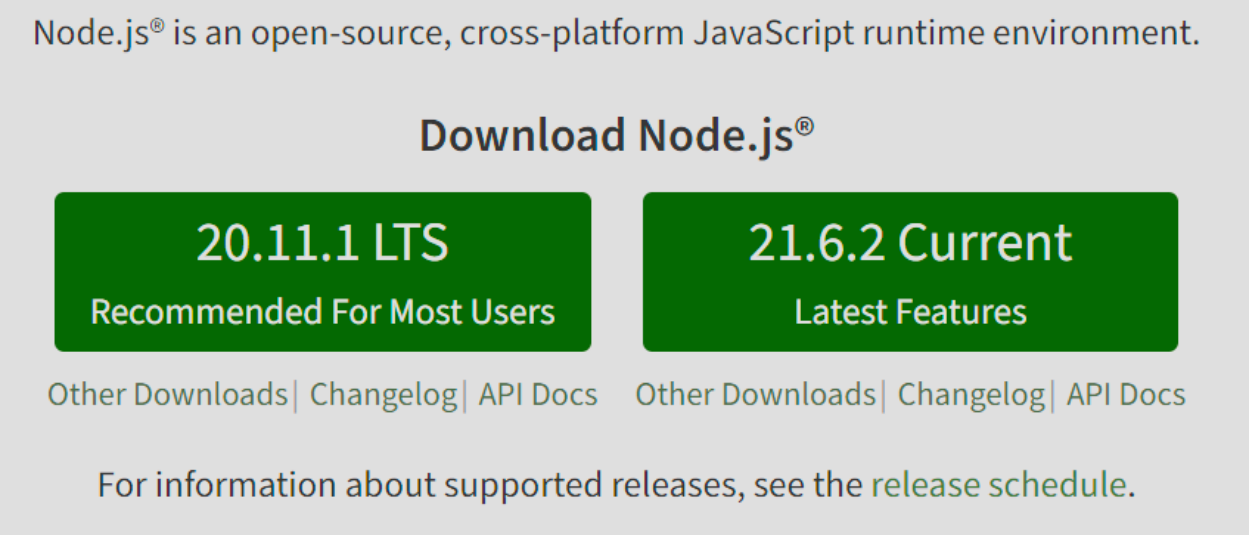
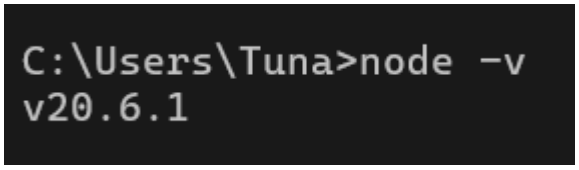
1. **Hướng dẫn cài NodeJS**

- Truy cập vào liên kết: https://nodejs.org/en

- Chọn vào phần Recomended For Most Users để tải về file exe



- Tiến hành cài đặt file vừa tải về: Auto Next
- Kiểm tra nodeJS đã được cài đặt chưa bằng cách mở terminal và nhập: node -v



- Hiển thị version thì thành công bước cài đặt nodeJS

2. Hướng dẫn tạo project với nodeJS(Sử dụng npm)

- Tạo một folder vd: cnm_lab04_mvc
- Mở folder trên vscode, mở terminal của vscode bằng phím tắt: Ctrl + Shift + `
- Nhập: **npm init -y**
- Nhấn enter để cài đặt file package.json. Sau khi hoàn thành trong folder sẽ hiển thị 1 file package.json

```
{
"name": "lab_04_mvc",
"version": "1.0.0",
"description": "",
"main": "index.js",
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
},
"author": "",
"license": "ISC"
}
```

- Thêm **"start": "node index"** vào thẻ scripts trong file package.json
- Tạo 1 file **index.js** ở root folder với nội dung console.log('hi')
- Sau đó nhập **npm star**t trong terminal và nhấn enter. Terminal hiển thị hi -> OK

3. Hướng dẫn cài đặt các packages cho ứng dụng

- Ở terminal nhập: **npm install ejs nodemon express**. Đợi khoảng một chút để node cài đặt packages. Các packages được cài đặt sẽ được hiển thị ở phần dependences trong package.json
- Dùng nodemon để tự động cập nhật code mới trong khi chạy. Vào package.json. Đổi **"start": "nodemon index.js"**.
- Bạn thử đổi hi ở file **index.js** và lưu code. Tác dụng của nodemon đấy!

4. Cấu hình server:

- Đổi nội dung của file **index.js** thành:
- Trong đó:
    - app.use(): dùng để cài đặt các middlewares cho ứng dụng
    - app.set(): cấu hình ứng dụng express ở đây sẽ cấu hình view để hiển thị các file ejs ở phần view
    - Phần routers thì sẽ dùng chủ yếu 2 phương thức GET, POST (vì form ở html hiện tại chỉ hỗ trợ 2 phương thức này). Còn các phương thức PUT, PATCH, DELETE. Vui lòng tìm hiểu RESTFul API.

JS index.js > ...

```js
const express = require('express');
const PORT = 3000;
const app = express();
let courses = require('./data');

//register middleware
app.use(express.urlencoded({ extended: true }));
app.use(express.static('./views')); // Cho phép dùng các tài nguyên tĩnh như css, javascript, images,...

//config view
app.set('view engine', 'ejs'); // Khai báo rằng app sẽ dùng engine EJS để render trang web
app.set('views', './views'); // Nội dung render trang web sẽ nằm trong thư mục tên `views`

app.get('/', (req, resp) => {
    return resp.render('index', { courses }) //send data to ejs
});


app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

- Tạo file **data.js** để chứa mảng data courses

```
JS data.js
JS data.js > courses
const courses = [
    {
        id: 1,
        name: 'Cơ sở dữ liệu',
        course_type: 'Cơ sở',
        semester: 'HK1-2020-2021',
        department: 'K.CNTT'
    },
    {
        id: 2,
        name: 'Cấu trúc dữ liệu',
        course_type: 'Cơ sở',
        semester: 'HK1-2020-2021',
        department: 'K.CNTT'
    },
    {
        id: 3,
        name: 'Công nghệ phần mềm',
        course_type: 'Cơ sở ngành',
        semester: 'HK1-2020-2021',
        department: 'K.CNTT'
    },
    {
        id: 4,
        name: 'Công nghệ mới',
        course_type: 'Chuyên ngành',
        semester: 'HK1-2020-2021',
        department: 'K.CNTT'
    },
    {
        id: 5,
        name: 'Đồ án môn học',
        course_type: 'Chuyên ngành',
        semester: 'HK1-2020-2021',
        department: 'K.CNTT'
    }
];

module.exports = courses;// Xuất data để dùng ở nơi khác
```

5. Cấu hình views:

- Tạo folder views ở root folder. Bên trong tạo 2 file **index.ejs** và **index.css**
- Nội dung file **index.ejs**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="index.css">
</head>
<body>
    <div class="content">
        <table>
            <caption>Danh sách các môn học</caption>

            <thead>
                <tr>
                    <th>STT</th>
                    <th>Tên môn học</th>
                    <th>Loại môn học</th>
                    <th>Học kỳ</th>
                    <th>Khoa</th>
                </tr>
            </thead>

            <tbody>
                <% courses.forEach(item => { %>
                    <tr>
                        <td><%= item.id %></td>
                        <td><%= item.name %></td>
                        <td><%= item.course_type %></td>
                        <td><%= item.semester %></td>
                        <td><%= item.department %></td>
                    </tr>
                <% }); %>
            </tbody>
        </table>
    </div>
</body>
</html>
```

- Nội dung file **index.css**:

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
}

table {
    border-collapse: collapse;
    width: 100%;

    caption {
        padding: 4px;
        margin: auto;
        font-size: 24px;
        font-weight: 700;
        color: red;
        background-color: gray;
        width: fit-content;
    }
}

th, td {
    border: 1px solid #dddddd;
    text-align: left;
    padding: 8px;
}

th {
    background-color: rgb(84, 247, 84);
}
```

- Chạy **npm start** để xem kết quả.

6. Chức năng thêm item mới và xóa:

- Tại trang index.ejs, tạo 1 form để nhập dữ liệu.

- Thuộc tính action của form sẽ trỏ tới api http://localhost:3000/save với method post.

```
<form action="/save" method="post">
    <input type="text" name="id" placeholder="Nhập id.." />
    <br>
    <input type="text" name="name" placeholder="Nhập name.." />
    <br>
    <input type="text" name="course_type" placeholder="Nhập course_type.." />
    <br>
    <input type="text" name="semester" placeholder="Nhập semester.." />
    <br>
    <input type="text" name="department" placeholder="Nhập department.." />
    <br>
    <button type="submit">Submit</button>
</form>
```

- Thêm router /save mới phương thức post ở phía backend:

```
app.post('/save', (req, resp) => {
    const id = Number(req.body.id);
    const name = req.body.name;
    const course_type = req.body.course_type;
    const semester = req.body.semester;
    const department = req.body.department;

    const params = {
        "id": id,
        "name": name,
        "course": course_type,
        "semester": semester,
        "department": department
    }

    courses.push(params);

    return resp.redirect('/');
});
```

- Thêm router /delete với phương thức post:

```
app.post('/delete', (req, res) => {
    const listCheckboxSelected = Object.keys(req.body);// Lấy ra tất cả checkboxes
    //req.body trả về 1 object chứa các cặp key & value định dạng:
    // '123456': 'on',
    // '123458': 'on',
    //listCheckboxSelected trả về 1 array: [ '123456', '123458', '96707133' ]
    if(listCheckboxSelected.length <= 0){
        return res.redirect('/');
    }

    function onDeleteItem(length){// Định nghĩa hàm đệ quy xóa
        const maSanPhamCanXoa = Number(listCheckboxSelected[length]);// Lấy ra maSP cần xóa

        data = data.filter(item => item.maSanPham !== maSanPhamCanXoa);// Dùng hàm filter hoặc .split, hoặc nhiều cách khác để xóa mảng.
        if(length > 0) {
            console.log('Data deleted:: ', JSON.stringify(data));
            onDeleteItem(length - 1);
        } else // Nếu length = 0 tức là không còn gì trong listCheckBox để xóa -> Render lại trang index để cập nhật data.
            return res.redirect('/');
    }
    onDeleteItem(listCheckboxSelected.length - 1);// Gọi hàm đệ quy xóa
});
```