TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
BỘ MÔN TIN HỌC ỨNG DỤNG

Lớp 16Y1A1 – Ngành Công nghệ thông tin – K42
Niên Luận Cơ Sở - CT270

ĐỀ TÀI

TÌM HIỂU VÀ XÂY DỰNG RESTFUL API WEB SERVICE THEO MÔ HÌNH MERN STACK

GIÁO VIÊN HƯỚNG DẪN:
Th.S Hồ Văn Tú

SINH VIÊN THỰC HIỆN:
Bành Ngọc Thụy Thảo
MSSV: B1606935

Cần Thơ, Tháng 12/2019

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành đến gia đình đã động viên em trong suốt quá trình học tập.

Em xin cảm ơn Bộ môn Tin học ứng dụng, khoa Công nghệ thông tin và Truyền thông, Trường Đại học Cần Thơ đã tạo điều kiện tốt nhất để em thực hiện đề tài này.

Dưới sự hướng dẫn của ThS. Hồ Văn Tú, em đã hoàn thành đề tài "Tìm hiểu và xây dựng RESTful API web service theo mô hình MERN Stack". Em xin bày tỏ lời cảm ơn sâu sắc đến thầy vì đã tận tình giúp đỡ, định hướng cho em trong suốt thời gian qua.

Trong quá trình thực hiện, em đã tích lũy được thêm nhiều kiến thức, giúp bản thân em dần hoàn thiện hơn. Đồng thời, kết hợp với kiến thức nền tảng đã có trong thời gian học tập những năm qua, em đã hoàn thành đề tài này.

Em rất mong nhận được sự góp ý, nhận xét từ thầy để đề tài được hoàn thiện hơn.

Cuối lời, em xin chúc thầy dồi dào sức khỏe, hạnh phúc và đạt được nhiều thành công trong công việc và cuộc sống.

Trân trọng!

Cần Thơ, ngày 05 tháng 12 năm 2019

Sinh viên thực hiện

Bành Ngọc Thụy Thảo

NHẬN XÉT CỦA CÁN BỘ HƯỚNG DẪN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Cần Thơ, ngày … tháng … năm 2019
Cán bộ hướng dẫn

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Ký hiệu, chữ viết tắt	Diễn giải
API	Application Program Interface
HTTP	HTTP HyperText Transfer Protocol
JSON	JavaScript Object Notation
MERN	MongoDB - Express - React - NodeJS
MVC	Model - View - Controller
REST	REpresentational State Transfer
URL	Uniform Resource Locator
XML	eXtensible Markup Language

TÓM TẮT

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, các công nghệ web cũng theo đó mà thay đổi liên tục. Việc tìm hiểu, nắm vững và áp dụng những công nghệ đó vào việc xây dựng website là điều vô cùng cần thiết đối với mỗi lập trình viên. Đó cũng là lý do đề tài "Tìm hiểu và xây dựng RESTFul API web service theo mô hình MERN Stack" được thực hiện.

Dịch vụ web, hay còn gọi là web service, xây dựng theo chuẩn RESTFul API có nhiều ưu điểm so với những website truyền thống. Việc xây dựng một web service theo mô hình MERN Stack (MongoDB, Express, React và NodeJS), bộ các công nghệ mã nguồn mở của Javascript, sẽ giúp cho website nâng cao hiệu suất và đảm bảo tính an toàn.

Web service sẽ được xây dựng bằng NodeJS và sẽ kết hợp với MongoDB, hệ quản trị cơ sở dữ liệu có khả năng lưu trữ và xử lý dữ liệu rất tốt. Đây như một bộ đôi tương thích với nhau cho một hệ thống lớn. React sẽ được sử dụng vào việc xây dựng giao diện của website. Công cụ hỗ trợ lập trình được sử dụng bao gồm Visual Code, Postman và MongoDB Compass Community.

Đề tài thực hiện gồm hai nội dung chính. Phần đầu tiên sẽ trình bày lý thuyết các khái niệm được nhắc bên trên và phần thứ hai sẽ trình bày các bước để xây dựng một web service theo mô hình MVC thông qua việc xây dựng trang web quản lý thông tin sinh viên, môn học, lớp học phần với những chức năng cơ bản.

Từ khóa: *Web service, RESTFul API, MongoDB, Express, React, NodeJS, MVC*

ABSTRACT

In the era where information technology is advancing at rapid speed, web techniques are also constantly changing. Learning, mastering and applying them in building a website is very necessary for every programmer or developer. For that reason, the project "Learning and building a RESTful API web service based on MERN Stack model" is realized.

Web service built in RESTful API standard has more advantages than other traditional websites. Creating a web service based on MERN Stack model (MongoDB, Express, React and NodeJS), a set of open source technologíes of Javascript, allow the website to improve the efficiency and guarantee its security.

Web service will be built by NodeJS and combined with MongoDB, a database management system that can store and process data in an effective way. This is considered as a duo compatible for a large system. React will be applied to create the website interface. Visual Code, Postman and MongoDB Compass Community will be used as programming support tools.

This project includes two main parts. The first one will present all concepts above and the second one will show steps to build a web service in MVC model by creating a website for management students information, subjects, classes subjects with fundamental functions.

Keywords: *Web service, RESTFul API, MongoDB, Express, React, NodeJS, MVC*

CHƯƠNG 1: TỔNG QUAN

1.1. Đặt vấn đề

Website, hay các trang web là một từ khóa rất quen thuộc đối với tất cả mọi người trong thời đại ngày nay. Cùng với sự phát triển nhanh chóng của công nghệ web, rất nhiều ngôn ngữ lập trình, thư viện, framework đã được sáng tạo ra, nhằm nâng cao hiệu suất của trang web cũng như cải thiện về mặt giao diện, giúp người dùng có những trải nghiệm trực quan hơn và những nhà lập trình cũng dễ dàng hơn trong việc phát triển.

Bên cạnh đó, dịch vụ web (web service) xây dựng theo mô hình MERN Stack (MongoDb – Express – React – NodeJS) là những công nghệ open source liên quan khá nổi tiếng hiện nay. Việc tìm hiểu, nắm rõ những công nghệ này và áp dụng nó là một điều rất cần thiết đối với lập trình viên. Chính vì lý do đó, em đã thực hiện đề tài "Tìm hiểu và xây dựng web service RESTful API theo mô hình MERN Stack".

1.2. Mục tiêu đề tài

Mục tiêu chính của đề tài là tìm hiểu và nắm được các khái niệm về web service, RESTful API, cách sử dụng MongoDB, Express, React, NodeJS và các thư viện, framework liên quan. Từ đó có thể áp dụng và xây dựng một web service căn bản theo mô hình MVC và thiết kế các API theo chuẩn RESTful.

1.3. Phương pháp nghiên cứu

Đầu tiên, em sẽ tìm hiểu về mặt lý thuyết cũng như cách hoạt động của Web service, MongoDB, Express, React và NodeJS. Và sau khi nắm rõ lý thuyết, sẽ bắt tay vào xây dựng một web service quản lý thông tin sinh viên, môn học và danh sách lớp học phần cũng như hiển thị tin tức từ một số trang web khác.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Web service

Web Service đã mở ra một hướng mới cho việc phát triển các ứng dụng Internet. Web Service có thể tạm hiểu là dịch vụ web. Web Service kết hợp sử dụng nhiều công nghệ khác nhau cho phép các ứng dụng khác nhau được viết bằng các ngôn ngữ khác nhau, vận hành trên các nền tảng khác nhau nhưng vẫn có thể trao đổi với nhau thông qua môi trường Internet.

Web Service là một khái niệm rộng hơn so với khái niệm web thông thường, nó cung cấp các thông tin thô, và khó hiểu với đa số người dùng, chính vì vậy nó được sử dụng bởi các ứng dụng. Các ứng dụng này sẽ chế biến các dữ liệu thô trước khi trả về cho người dùng cuối cùng. Các Web Service thường cung cấp các dữ liệu thô mà nó khó hiểu đối với đa số người dùng thông thường, chúng thường được trả về dưới dạng XML hoặc JSON.

Một Web Service được tạo nên bằng cách lấy các chức năng và đóng gói chúng sao cho các ứng dụng khác dễ dàng nhìn thấy và có thể truy cập đến những dịch vụ mà nó thực hiện, đồng thời có thể yêu cầu thông tin từ Web Service khác. Nó bao gồm các mô đun độc lập cho hoạt động cho các yêu cầu của Client và bản thân nó được thực thi trên server.[5]

2.2. RESTful API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

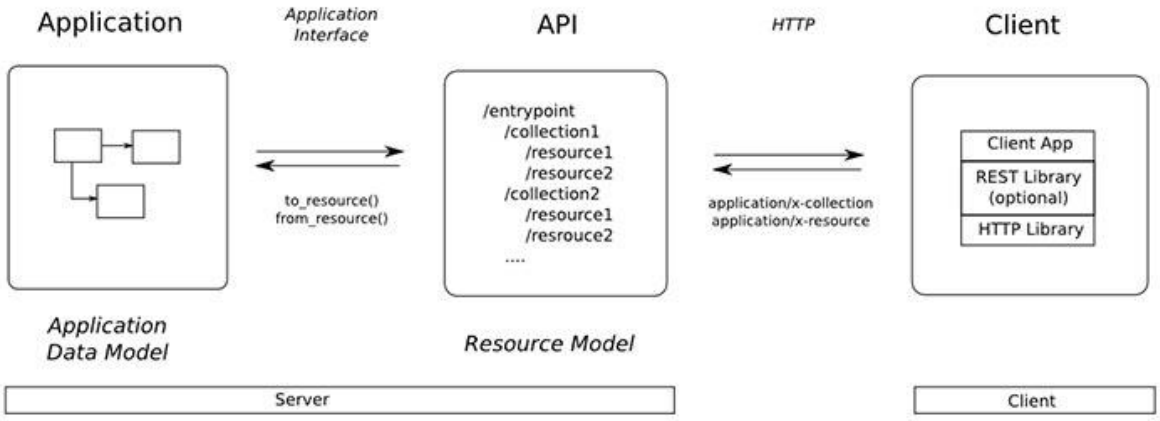
Diễn giải các thành phần :

- API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.
- REST (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một

số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, … đến một URL để xử lý dữ liệu.

- RESTful API là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.[17]

Cách thức hoạt động



Hình 2.1 Cách thức hoạt động của REST

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

- GET (SELECT): Trả về một tài nguyên hoặc một danh sách tài nguyên.
- POST (CREATE): Tạo mới một tài nguyên.
- PUT (UPDATE): Cập nhật thông tin cho tài nguyên.
- DELETE (DELETE): Xoá một tài nguyên.

Xác thực và dữ liệu trả về

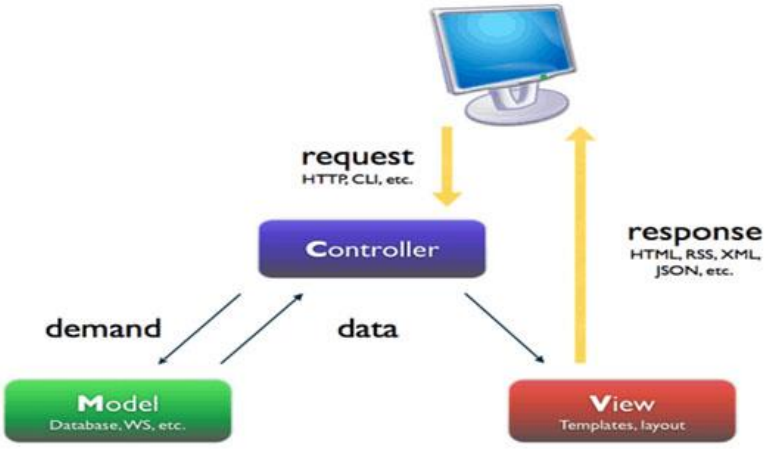
RESTful API không sử dụng session và cookie, nó sử dụng một access_token với mỗi request. Dữ liệu trả về thường có cấu trúc như sau:

```
{
   "data":{
      "id":"B1606935",
      "name":"Bành Ngọc Thụy Thảo"
   }
```

2.3. Mô hình MVC

MVC là viết tắt của Model – View – Controller. Đây là một kiến trúc phần mềm hay mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. Nó là mô hình phân bố source code thành 3 phần, mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.

- **Controller**: Giữ nhiệm vụ nhận điều hướng các yêu cầu từ người dùng và gọi đúng những phương thức xử lý chúng… Chẳng hạn thành phần này sẽ nhận request từ url và form để thao tác trực tiếp với Model.
- **Model**: Đây là thành phần chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các Class, hàm xử lý…
- **View:** Là nơi chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh… nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.



Hình 2.2 Cách thức hoạt động mô hình MVC

Khi có một yêu cầu từ phía client gửi đến server, Bộ phận controller có nhiệm vụ nhận yêu cầu, xử lý yêu cầu đó. Và nếu cần, nó sẽ gọi đến phần model, vốn là bộ

phận làm việc với Database. Sau khi xử lý xong, toàn bộ kết quả được đẩy về phần View. Tại View, sẽ render ra mã tml tạo nên giao diện, và trả toàn bộ html về trình duyệt để hiển thị.[13]

2.4. Các nền tảng phía Client

2.4.1. ReactJS

React là một thư viện Javascript, dùng để xây dựng giao diện người dùng và được duy trì bởi Facebook. React được để phát triển các ứng dụng single-page hoặc di động. Trong một ứng dụng theo mô hình MVC thì React chính là lớp View.[2]

2.4.1.1 React component

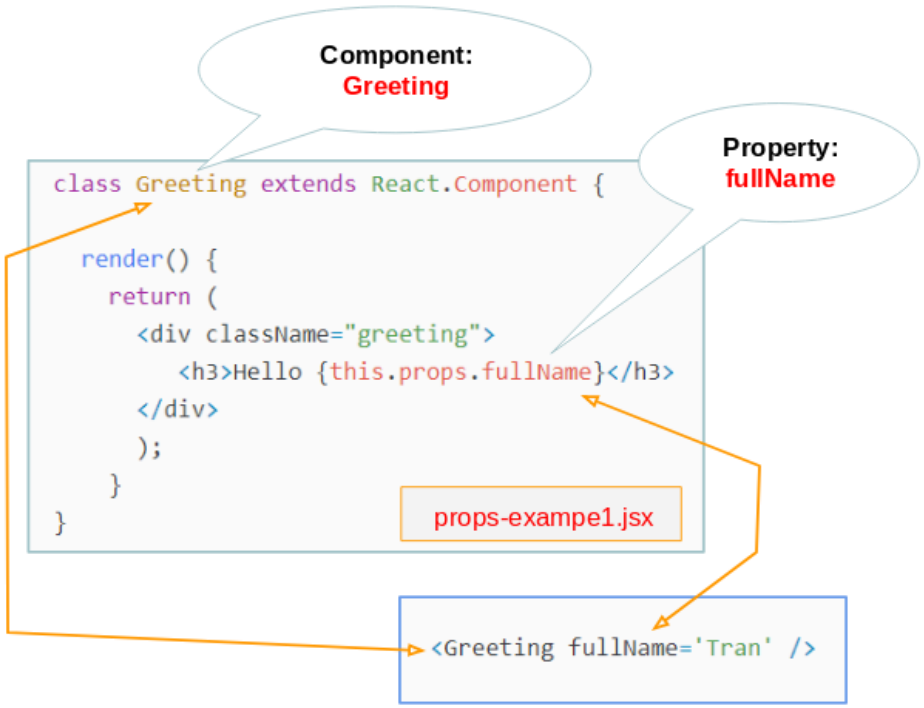
Trong ReactJS, mỗi đoạn code sẽ được phân chia thành những Component không lệ thuộc lẫn nhau và có thể tái sử dụng khi cần thiết. Các component sử dụng phương thức render() để trả về nội dung được hiển thị. Dưới đây là ví dụ cơ bản về việc sử dụng một component.[15]

```
Import React, {Component} from 'react';
class HelloMessage extends Component {
  render() {
    return (
      <div>
        Hello World!
      </div>
    );
  }
}

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

2.4.1.2 Props

Trong ReactJS khi tạo một Component, giống như việc tạo ra một thẻ (tag) mới. Mỗi property của Component sẽ tương ứng với một attribute của thẻ.[10]



Hình 2.3 Minh họa về props

2.4.1.3 State

Giống như props, state cũng giữ thông tin về component. Tuy nhiên, loại thông tin và cách xử lý nó khác nhau. State hoạt động khác với Props. State là thành phần của component, trong khi các props lại được truyền giá trị từ bên ngoài vào component. [10]

```
class Today extends Component {
  constructor(props) {
    super(props);
    this.state = { date: new Date() };
  }
  render() {
    return (
      <div>
        <p>Today is {this.state.date.toLocaleDateString()} </p>
      </div>
    );
  }
}
export default Today;
```

Hình 2.4 Minh họa về state

2.4.1.4 Lifecycle

Trong React, để override một số nhiệm vụ, các React lifecycle methods – các phương thức trong vòng đời của React component sẽ được sử dụng.

Các phương thức trong React Component Lifecycle có thể chia ra làm 3 pha chính là: Mounting, Updating và Unmounting.

- **Các phương thức trong pha Mounting**

Mounting là giai đoạn khi React Component được tạo ra và render lên trên DOM tree.

Các React lifecycle methods được gọi trong giai đoạn này lần lượt là:

+ **constructor()** : Đối với class nói chung, constructor() luôn là phương thức được gọi đến đầu tiên mỗi khi khởi tạo. Tuy nhiên, chỉ nên sử dụng phương thức này với 2 mục đích: khởi tạo state cho component và khai báo phương thức với this.
+ **static getDerivedStateFromProps()** : phương thức này được gọi ngay trước khi element được render trong DOM.
+ **render() :** Phương thức này dùng để miêu tả cấu trúc của Component sau khi nó được chèn vào DOM tree. Nó bắt buộc được gọi lần đầu tiên để chèn Component vào HTML, và có thể được gọi lại để cập nhật giao diện mỗi khi state của Component thay đổi.
+ **componentDidMount() :** phương này được thực thi khi 1 component được render trên client side. Đây là nơi các hàm AJAX requests, DOM hoặc update state được thực thi.

- **Các phương thức trong pha Updating**

Pha tiếp theo trong vòng đời React Component là khi một component được cập nhật. Một component được cập nhật khi có sự thay đổi trong state hoặc props của nó.

Các React lifecycle methods được gọi trong giai đoạn này lần lượt là:

+ **static getDerivedStateFromProps() :** phương thức này được gọi ngay trước khi element được render trong DOM.
+ **shouldComponentUpdate()** : sẽ trả về kết quả true or false. Phương thức này sẽ xác định 1 component có được cập nhật hay không. Mặc định giá trị này là true. Nếu không muốn component render lại sau khi cập nhật state hay props thì return giá trị thành false.

+ **render()** : phương thức này sẽ render lại HTML với giá trị mới vừa thay đổi.
+ **getSnapshotBeforeUpdate()** : cho phép truy cập vào props và state trước khi cập nhật
+ **componentDidUpdate()** : Phương thức này được gọi sau khi việc update kết thúc - component với những dữ liệu mới đã được cập nhật xong lên giao diện.

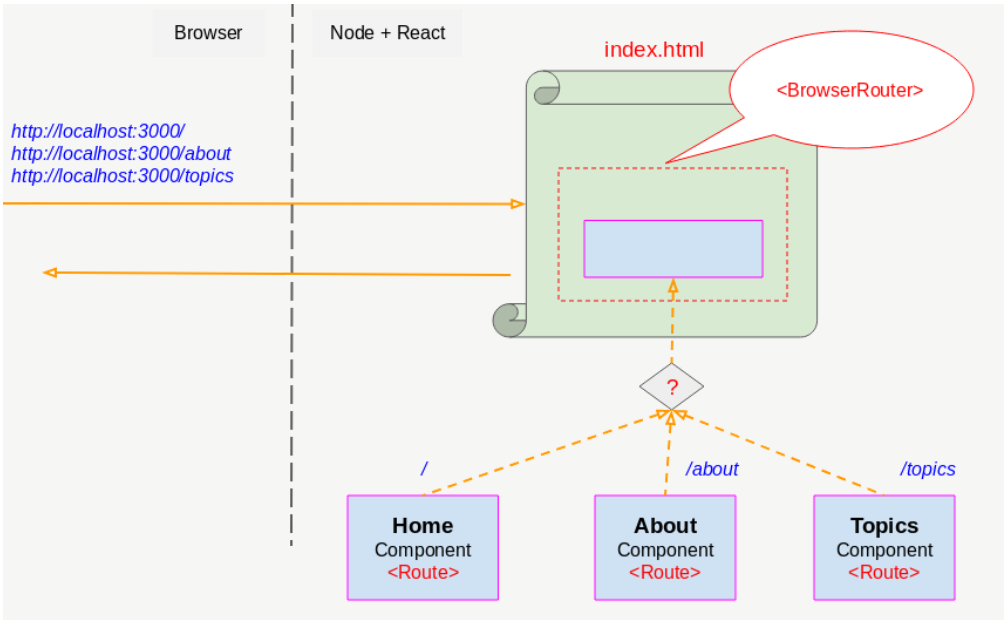
- **Các phương thức trong pha Unmouting**

Unmounting là giai đoạn khi React Component bị xoá khỏi DOM tree.

Trong giai đoạn này, chỉ có một phương thức được gọi duy nhất là: componentWillUnmount(). Phương thức này được gọi khi component chuẩn bị xóa khỏi DOM.[11]

2.4.1.5 React Router

React Router là một thư viện định tuyến (routing) tiêu chuẩn trong React. Nó giữ cho giao diện của ứng dụng đồng bộ với URL trên trình duyệt. React Router cho phép định tuyến "luồng dữ liệu"(data flow) trong ứng dụng một cách rõ ràng.[3]



Hình 2.5 Minh họa về React Router

Một số component phổ biến trong React Router:

- **BrowserRouter**: sử dụng history API trong HTML5 để theo dõi lịch sử bộ định tuyến
- **Route**: Định nghĩa một ánh xạ (mapping) giữa một URL và một Component. Điều đó có nghĩa là khi người dùng truy cập theo một URL trên trình duyệt, một Component tương ứng sẽ được render trên giao diện.

```
<Router>
    <div className="App">
        <Route path="/" exact component={Home} />
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
        <Route component={NotFound}/>
    </div>
</Router>
```

Trong đó:

+ path: Là đường dẫn trên URL.
+ exact: Giúp cho route này này chỉ hoạt động nếu URL trên trình duyệt phù hợp tuyệt đối với giá trị của thuộc tính path của nó.
+ component: là component sẽ được render ra tương ứng với Route đó.

- **Link** : Trong HTML thì cặp thẻ để chuyển hướng đó là thẻ <a> thì trong React sẽ sử dụng cặp thẻ <Link></Link>

```
<Link to="/about">About</Link>
```

Trong đó:

+ to: Giống như thuộc tính href trong thẻ a.

- **NavLink** : giống với Link về cách sử dụng, nhưng NavLink tốt hơn vì nó hỗ trợ thêm một số thuộc tính như là activeClassName và activeStyle. Hai thuộc tính này giúp cho khi mà nó trùng khớp thì nó sẽ được active lên và có thể style cho nó.

```
<NavLink to="/" exact activeStyle={{ fontWeight: "bold" }}>
   Tin tức
</NavLink>
```

- **Redirect**: Chức năng dùng để chuyển trang, có thể truy xuất thông tin trang trước đó thông qua đối tượng location.[16]

```
<Redirect to="/" />
```

2.4.2. HTML

HTML là chữ viết tắt của cụm từ HyperText Markup Language (dịch là Ngôn ngữ đánh dấu siêu văn bản) được sử dụng để tạo một trang web, trên một website có thể sẽ chứa nhiều trang và mỗi trang được quy ra là một tài liệu HTML (thi thoảng mình sẽ ghi là một tập tin HTML). Cha đẻ của HTML là Tim Berners-Lee, cũng là người khai sinh ra World Wide Web và chủ tịch của World Wide Web Consortium (W3C – tổ chức thiết lập ra các chuẩn trên môi trường Internet).[8]

2.4.3. CSS

Trong tin học, các tập tin định kiểu theo tầng – dịch từ tiếng Anh là Cascading Style Sheets (CSS) – được dùng để miêu tả cách trình bày các tài liệu viết bằng ngôn ngữ HTML và XHTML. Ngoài ra ngôn ngữ định kiểu theo tầng cũng có thể dùng cho XML, SVG, XUL. Các đặc điểm kỹ thuật của CSS được duy trì bởi World Wide Web Consortium.[4]

2.4.4. Javascript

Theo phiên bản hiện hành, JavaScript là một ngôn ngữ lập trình kịch bản dựa trên đối tượng được phát triển từ các ý niệm nguyên mẫu. Ngôn ngữ này được dùng rộng rãi cho các trang web, nhưng cũng được dùng để tạo khả năng viết script sử dụng các đối tượng nằm sẵn trong các ứng dụng.[9]

2.4.5. Bootstrap

Bootstrap là một framework cho phép thiết kế website responsive nhanh hơn và dễ dàng hơn. Bootstrap là bao gồm các HTML templates, CSS templates và Javascript tao ra những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác. Trong bootstrap có thêm các plugin Javascript trong nó. Giúp cho việc thiết kế responsive dễ dàng hơn và nhanh chóng hơn.[1]

2.5. Các nền tảng phía server

2.5.1. MongoDB

Định nghĩa

MongoDB là một cơ sở dữ liệu mã nguồn mở và là cơ sở dữ liệu NoSQL hàng đầu, được hàng triệu người sử dụng. MongoDB được viết bằng C++.

NoSQL là 1 dạng CSDL mã nguồn mở không sử dụng Transact-SQL để truy vấn thông tin. NoSQL viết tắt bởi: None-Relational SQL, hay có nơi thường gọi là Not-

Only SQL. CSDL này được phát triển trên Javascript Framework với kiểu dữ liệu JSON. (Cú pháp của JSON là "key:value") NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu xót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS về tốc độ, tính năng, khả năng mở rộng, memory cache,...

Các thuật ngữ trong MongoDB

_id – Là trường bắt buộc có trong mỗi document. Trường _id đại diện cho một giá trị duy nhất trong document MongoDB. Trường _id cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một _id đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB.

Collection – Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.

Cursor – Đây là một con trỏ đến tập kết quả của một truy vấn. Máy khách có thể lặp qua một con trỏ để lấy kết quả.

Database – Nơi chứa các Collection, giống với cơ sở dữ liệu RDMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một máy chủ MongoDB có thể chứa nhiều Database.

Document – Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị.

Field – Là một cặp name – value trong một document. Một document có thể có không hoặc nhiều trường. Các trường giống các cột ở cơ sở dữ liệu quan hệ.

JSON – Viết tắt của JavaScript Object Notation. Con người có thể đọc được ở định dạng văn bản đơn giản thể hiện cho các dữ liệu có cấu trúc. Hiện tại JSON đang hỗ trợ rất nhiều ngôn ngữ lập trình.

Index – Là những cấu trúc dữ liệu đặc biệt, dùng để chứa một phần nhỏ của các tập dữ liệu một cách dễ dàng để quét. Chỉ số lưu trữ giá trị của một fields cụ thể hoặc thiết lập các fields, sắp xếp theo giá trị của các fields này. Index hỗ trợ độ phân tích một cách hiệu quả các truy vấn. Nếu không có chỉ mục, MongoDB sẽ phải quét tất cả các documents của collection để chọn ra những document phù hợp với câu truy vấn. Quá trình quét này là không hiệu quả và yêu cầu MongoDB để xử lý một khối lượng lớn dữ liệu.[12]

```
_id: ObjectId("5db2843b54494c2c98803bbb")
id: "TN202"
name: "Lập trình web 6"
description: "Giúp người học có thể xây dựng một trang web với giao diện và chức năn..."
__v: 0
```

Hình 2.6 Một document trong MongoDB

2.5.2. Express

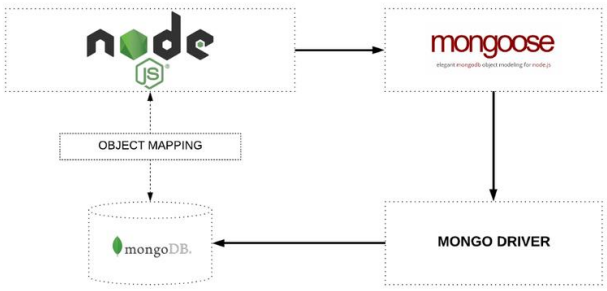
Express là một framework nhỏ nhưng linh hoạt, được xây dựng trên nền tảng NodeJS. Nó cung cấp nhiều tính năng mạnh mẽ để phát triển web và ứng dụng di động. Express có nhiều package hỗ trợ lập trình. Một số chức năng chính của express là thiết lập các lớp trung gian để trả về các request, định nghĩa router cho phép sử dụng với cá hành động khác nhau dựa trên phương thức HTTP và URL, cho phép trả về các trang HTML dựa vào các tham số.[6]

2.5.3. NodeJS

NodeJS là một mã nguồn mở được xây dựng trên nền tảng Javascript V8 Engine. Nodejs sử dụng rộng bởi hàng ngàn lập trình viên trên toàn thế giới. Nó có thể chạy trên nhiều nền tảng hệ điều hành như Windows, Linux, MacOS. NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất. Đặc tính nổi trội của Nodejs là tính bất đồng bộ. Điều này giúp các request được xử lý ngay lập tức.[14]

2.5.4. Mongoose

Mongoose là một thư viện mô hình hóa đối tượng (Object Data Model - ODM) cho MongoDB và Node.js. Nó quản lý mối quan hệ giữa dữ liệu, cung cấp sự xác nhận giản đồ và được sử dụng để dịch giữa các đối tượng trong mã và biểu diễn các đối tượng trong MongoDB.[7]



Hình 2.7 Cách thức hoạt động Mongoose

CHƯƠNG 3: NỘI DUNG VÀ KẾT QUẢ THỰC HIỆN

3.1. Thiết kế cơ sở dữ liệu

Bảng 3.1 Bảng thực thể sinh viên

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	**SV_ID**	Integer	Mỗi sinh viên sẽ có một mã số riêng biệt Kiểu dữ liệu : số
2	**SV**_NAME	String	Họ và tên sinh viên Kiểu dữ liệu : chữ
3	**SV**_GENDER	String	Giới tính sinh viên Kiểu dữ liệu : chữ
4	**SV**_PHONE	String	Số điện thoại sinh viên Kiểu dữ liệu : chữ
5	**SV**_EMAIL	String	Email sinh viên Kiểu dữ liệu: chữ
6	**SV**_ADDRESS	String	Thông tin địa chỉ sinh viên Kiểu dữ liệu: chữ

Bảng 3.2 Bảng thực thể admin

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	**ADMIN_ID**	Integer	Mỗi admin sẽ có một mã số riêng biệt Kiểu dữ liệu : số
2	ADMIN_NAME	String	Họ và tên admin Kiểu dữ liệu: chữ

3	ADMIN_PASSWORD	String	Mật khẩu đăng nhập hệ thống admin Kiểu dữ liệu: chữ
4	ADMIN_USERNAME	String	Tên đăng nhập hệ thống admin Kiểu dữ liệu: chữ

Bảng 3.3 Bảng thực thể môn học

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	**MH_ID**	Integer	Mỗi môn học sẽ có một mã số riêng biệt Kiểu dữ liệu: số
2	MH_NAME	String	Tên môn học Kiểu dữ liệu: chữ
3	MH_DESCRIPTION	String	Mô tả mục tiêu, yêu cầu của môn học, Kiểu dữ liệu: chữ

Bảng 3.4 Bảng thực thể học phần

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	**HP_ID**	Integer	Mỗi học phần sẽ có một mã số riêng biệt Kiểu dữ liệu: số
2	MH_ID (Khóa ngoại)	Integer	Mã môn học mà lớp học phần này giảng dạy Kiểu dữ liệu: chữ
3	HP_TIME	String	Thời gian tiết học học phần diễn ra

			VD: tiết 1,2,3
4	HP_DATE	String	Ngày học học phần giảng dạy VD: thứ 2, 3, 4

3.2. Cài đặt

3.2.1. Thiết kế API server cơ sở dữ liệu theo chuẩn RESTful

Server cơ sở dữ liệu chạy trên cổng 4000

Bảng 3.5 Bảng danh sách API

STT	Địa chỉ	Phương thức	Chức năng
1	/sinhvien	GET	Lấy tất cả thông tin sinh viên
		POST	Thêm mới 1 thông tin sinh viên
2	/sinhvien/{id}	GET	Lấy thông tin sinh viên có id tương ứng
		PUT	Cập nhật thông tin sinh viên có id tương ứng
		DELETE	Xóa thông tin sinh viên có id tương ứng
3	/monhoc	GET	Lấy tất cả thông tin môn học
		POST	Thêm mới 1 thông tin môn học
4	/monhoc/{id}	GET	Lấy thông tin môn học có id tương ứng
		PUT	Cập nhật thông tin môn học có id tương ứng
		DELETE	Xóa thông tin môn học có id tương ứng
5	/admin	GET	Lấy tất cả thông tin admin
		POST	Thêm mới 1 thông tin admin
6	/admin/{id}	GET	Lấy thông tin admin có id tương ứng

		PUT	Cập nhật thông tin admin có id tương ứng
		DELETE	Xóa thông tin admin có id tương ứng
7	/admin/login	POST	Trả về thông tin admin có password và username tương ứng
8	/hocphan	GET	Lấy tất cả thông tin học phần
		POST	Thêm mới 1 thông tin học phần
9	/hocphan/{id}	GET	Lấy thông tin học phần có id tương ứng
		PUT	Cập nhật thông tin học phần có id tương ứng
		DELETE	Xóa thông tin học phần có id tương ứng
10	/tintuc/htql	GET	Lấy dữ liệu tin tức từ trang https://htql.ctu.edu.vn/htql/login.php
11	/tintuc/ctu	GET	Lấy dữ liệu tin tức từ trang https://www.ctu.edu.vn/thong-bao.html
12	/tintuc/baogiaoduc	GET	Lấy dữ liệu tin tức từ trang https://giaoduc.net.vn

3.2.2. Thiết kế route render component React

Server React chạy trên cổng 3000

Bảng 3.6 Bảng danh sách route render component React

STT	Địa chỉ	Chức năng
1	/	Render giao diện trang chủ
2	/danhsach/sinhvien	Render giao diện danh sách sinh viên
3	/danhsach/monhoc	Render giao diện danh sách môn học
4	/hocphan/danhsach	Render giao diện danh sách lớp học phần
4	/login	Render giao diện đăng nhập

3.2.3. Cài đặt phía Server

3.2.3.1 Khởi tạo thư mục server

Cài đặt các package: npm install express body-parser cors mongoose –save

Cấu trúc thư mục:

- Models: chứa các file js tạo model kết nối đến cơ sở dữ liệu
- Controllers: chứa các file js xử lý yêu cầu từ người dùng
- Routes: chứa các file js danh sách API
- File package.json: chứa thông tin cấu hình npm
- File package-lock.json: chứa thông tin các lần thay đổi của node modules
- Thư mục node_modules: chứa các module cho chương trình
- File server.js: chứa mã lệnh khởi tạo server
- File db.js: chứa mã lệnh kết nối đến cơ sở dữ liệu

3.2.3.2 Cài đặt server và kết nối đến cơ sở dữ liệu

Bên trong file server.js, thêm nội dung như sau:

```
// server.js
const express = require("express");
const app = express();
const bodyParser = require("body-parser");
const PORT = 4000;
const cors = require("cors");
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

app.listen(PORT, function() {
  console.log("Server is running on Port:", PORT);
});

//connect database
mongoose.Promise = global.Promise;
mongoose.connect(config.DB, { useNewUrlParser: true }).then(
  () => {
    console.log("Database is connected");
  },
  err => {
    console.log("Can not connect to the database" + err);
  }
);
```

Kết quả:

```
> api@1.0.0 start C:\Users\USER\Desktop\REACT\nlcs\api
> nodemon server.js

[nodemon] 1.19.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
(node:8808) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version
. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Server is running on Port: 4000
Database is connected
```

3.2.3.3 Khởi tạo model

Model sẽ khởi tạo một collection cho cơ sở dữ liệu Mongodb:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

let MonHoc = new Schema({
  id: { type: String },
  name: { type: String },
  description: { type: String }
});
module.exports = mongoose.model("MonHoc", MonHoc);
```

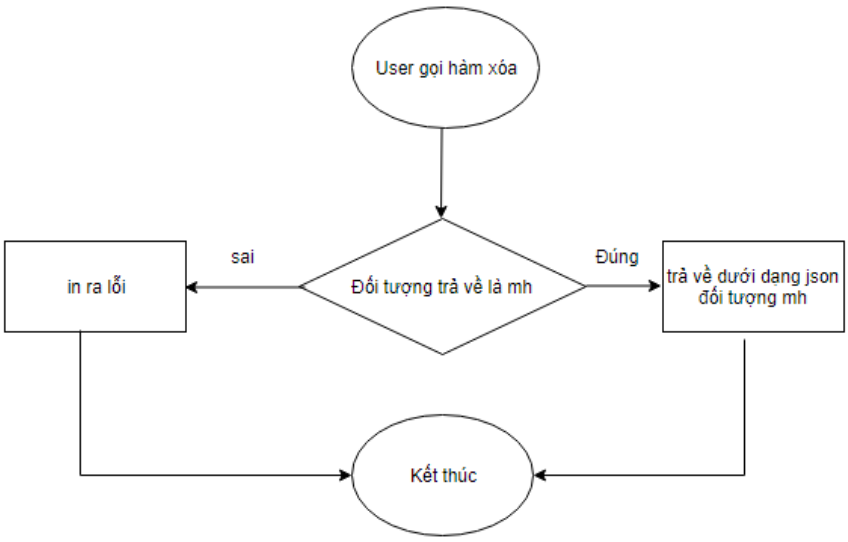
3.2.3.4 Khởi tạo controller

Các controller sẽ có nhiệm vụ thực hiện các yêu cầu mà người dùng gửi đến.

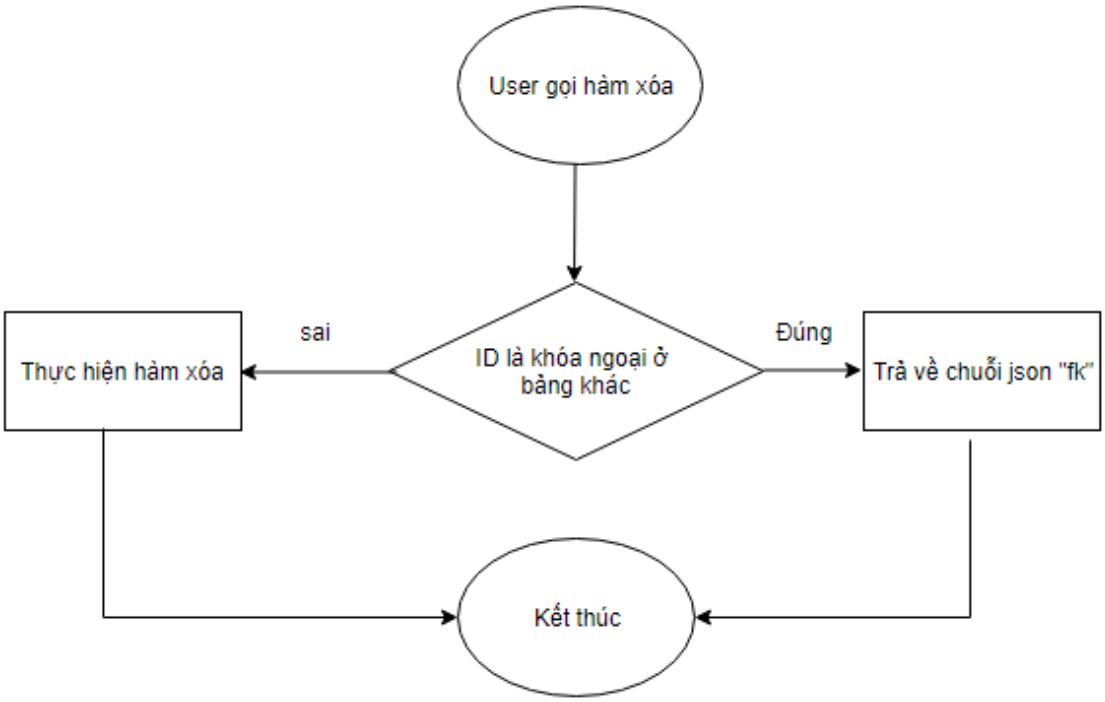
Tạo file monhoc-controller.js và viết các hàm:

- getListMonHoc : trả về danh sách môn học có trong cơ sở dữ liệu
- getMonHoc: trả về môn học có ID tương ứng
- addMonHoc: thêm mới môn học
- deleteMonHoc: xóa môn học
- updateMonHoc: cập nhật môn học

Chi tiết mã lệnh các hàm nằm trong phần mục lục.



Hình 3.1 Lưu đồ xử lý các hàm lấy thông tin, thêm, sửa môn học



Hình 3.2 Lưu đồ xử lý hàm xóa môn học

3.2.3.5 Khởi tạo controller web scraping

Sử dụng thư viện puppeteer của NodeJS để có thể tổng hợp dữ liệu HTML từ trang web khác.

Tạo file tintuc-controller.js và thêm nội dung:

```
const puppeteer = require("puppeteer");
const fs = require("fs");

exports.getDataCTU = (req, res) => {
  (async () => {
    const browser = await puppeteer.launch();
    const page = await browser.newPage();
    await page.goto("https://www.ctu.edu.vn/thong-bao.html");

    const results = await page.evaluate(() => {
      let items = document.querySelectorAll(".list-title a");
      let links = [];
      for (i = 0; i < 10; i++) {
        links.push({
          title: items[i].innerText,
          url: items[i].getAttribute("href")
        });
```

```
      }
      return links;
    });

    console.log(results);
    res.json(results);
  })();
};
```

Sau đó tạo route cho controller tin tức với phương thức GET tương tự theo hướng dẫn bên dưới.

3.2.3.6 Khởi tạo route

Các route sẽ có nhiệm vụ điều hướng các controller đúng theo yêu cầu được gửi đến.

Tạo file monhoc-routes.js và thêm nội dung:

```
const express = require("express");
const monhocRoutes = express.Router();
const monhocController = require("../controllers/monhoc-controller");
// Route lấy danh sách môn học
monhocRoutes.route("/").get(monhocController.getListMonHoc);
// Routes lấy dữ liệu môn học theo id
monhocRoutes.route("/:id").get(monhocController.getMonHoc);
// Route tạo mới dữ liệu
monhocRoutes.route("/").post(monhocController.addMonHoc);
// Route xóa dữ liệu
monhocRoutes.route("/:id").delete(monhocController.deleteMonHoc);
//  Route cập nhật dữ liệu
monhocRoutes.route("/:id").put(monhocController.updateMonHoc);
module.exports = monhocRoutes;
```

Bên file server.js, bổ sung nội dung:

```
//routes
const monhocRoute = require("./routes/monhoc-routes");
app.use("/monhoc", monhocRoute);
```

Để kiểm tra các API có hoạt động đúng không, ta có thể sử dụng phần mềm Postman để gửi request đến các API.

3.2.4. Cài đặt phía Client

3.2.4.1 Khởi tạo thư mục React

Cài đặt môi trường để chạy React: npm install -g create-react-app

Khởi tạo thư mục: npx create-react-app nlcs

Cài đặt module react-router-dom để thêm thư viện xử lý route: npm install react-router-dom –save

Cài đặt thư viện Axios để gửi request đến API: npm install axios --save

Cấu trúc thư mục :

- Thư mục public : chứa các file hình ảnh, file css,…
- Thư mục src: chứa các file khởi tạo component và các file xử lý phía client
- File package.json: chứa thông tin cấu hình npm
- File package-lock.json: chứa thông tin các lần thay đổi của node modules
- Thư mục node_modules: chứa các module cho chương trình

3.2.4.2 Khởi tạo các component

- **Tạo component AddForm**

Mô tả : gồm 1 nút Thêm và 1 modal. Khi nhấn vào nút thêm, sẽ hiển thị ra modal add form. Chi tiết mã lệnh nằm trong phần mục lục.

- **Tạo component MonHocEditForm**

Mô tả: Khi nhấn vào nút cập nhật ở mỗi dòng dữ liệu, sẽ hiện ra modal edit form. Chi tiết mã lệnh nằm trong phần mục lục.

- **Tạo component MonHoc**

Mô tả: Hiển thị danh sách các môn học. Component monhoc sẽ chứa component monhoc-table và component monhoc-table sẽ chứa component monhoc-item. Chi tiết mã lệnh nằm trong phần mục lục.

3.2.4.3 Hiển thị dữ liệu

Trong component monhoc, thêm nội dung cho phương thức componentDidMount để thực hiện lấy danh sách dữ liệu bằng cách gửi request với phương thức GET đến http://localhost:4000/monhoc.

```
componentDidMount() {
    api
      .get("/monhoc")
      .then(response => {
       this.setState({ items: response.data });
        })
      .catch(function(error) {
```

```
        console.log(error);
      });
  }
```

Danh sách dữ liệu trả về sẽ được lưu vào mảng items thông qua phương thức setStates và truyền qua component monhoc-table và monhoc-item. Mỗi dữ liệu sẽ tạo ra một component MonHocItem.

3.2.4.4 Tạo mới dữ liệu

Trong component monhoc-add-form, khởi tạo hàm onSubmit để thực hiện tạo mới dữ liệu bằng cách gửi request với phương thức POST đến http://localhost:4000/monhoc.

```
async onSubmit(e) {
    e.preventDefault();
    if (this.state.id === "") alert("Chưa nhập mã môn học!");
    else {
      const obj = {
        id: this.state.id,
        name: this.state.name,
        description: this.state.description
      };
      api.post("/monhoc", obj).then(res => console.log(res.data));
      this.setState({ id: "", name: "", description: "" });
      alert("Đã thêm thành công!");
      this.close();
      window.location.reload();
    }
  }
```

Đồng thời khai báo hàm vừa tạo trong constructor:

```
this.onSubmit = this.onSubmit.bind(this);
```

3.2.4.5 Cập nhật dữ liệu

Để thực hiện cập nhật, cần thực hiện hai công việc : hiển thị thông tin của đối tượng cần cập nhật vào form và thực hiện thao tác cập nhật.

Hiển thị thông tin

Để hiển thị thông tin, cần truyền được dữ liệu của môn học (gồm _id, id, name, description) của môn học từ component monhoc-item sang monhoc-table và đến monhoc.

Tại component monhoc-item, khởi tạo hàm handleEdit(item)

```
handleEdit(item) {
    this.props.onClickEdit(item);
}
```

và khai báo hàm trong constructor

```
    this.handleEdit = this.handleEdit.bind(this);
```

và gọi hàm trong phương thức onClick của nút Sửa

```
<button className="btn btn-info mr-3 btn-
sm" onClick={() => this.handleEdit(item)}>
        Sửa
</button>
```

Tại component monhoc-table, thêm props onClickEdit cho component MonHocItem

```
<MonHocItem
          key={index}
          item={item}
          index={index}
          onClickEdit={this.props.onClickEdit}
/>
```

Tại component môn học, bổ sung props onClickEdit cho component TableMonHoc

```
<TableMonHoc
          items={items}
          onClickEdit={this.handleEdit}
        />
```

và bổ sung nội dung cho hàm handleEdit(item).

```
handleEdit(item) {
    this.setState({ isShowEditForm: true, itemUpdated: item });
  }
```

State itemUpdated đã nhận được dữ liệu môn học cần cập nhật. Sau đó sẽ truyền giá trị này vào component MonHocEditForm bằng cách bổ sung props itemUpdated.

```
<MonHocEditForm
          onClickClose={this.handleClose}
          itemUpdated={this.state.itemUpdated}
          isShowEditForm={this.state.isShowEditForm}
/>
```

Tại component monhoc-edit-form, thực hiện việc gán dữ liệu cho các textbox tương ứng trong modal bằng cách dùng phương thức setState trong phương thức componentDidMount()

```
componentDidMount() {
    var item = this.props.itemUpdated;
    this.setState({
      _id: item._id,
      id: item.id,
      name: item.name,
      description: item.description
    });
  }
```

Thực hiện cập nhật

Trong component monhoc-edit-form, khởi tạo hàm onSubmit để thực hiện cập nhật dữ liệu bằng cách gửi request với phương thức PUT đến http://localhost:4000/monhoc/{id}

```
async onSubmit(e) {
    e.preventDefault();
    const obj = {
      id: this.state.id,
      name: this.state.name,
      description: this.state.description
    };
    api.put("/monhoc/" + this.state._id, obj).then(res => console.log(res.da
ta));
    await alert("Đã cập nhật thành công!");
    this.props.onClickClose();
    window.location.reload();
  }
```

Đồng thời khai báo hàm vừa tạo trong constructor:

```
this.handleEdit = this.handleEdit.bind(this);
```

3.2.4.6 Xóa dữ liệu

Để thực hiện việc xóa dữ liệu, cần lấy được id của đối tượng cần xóa và thực hiện thao tác xóa.

Lấy id của đối tượng

Để lấy id, cần truyền được id và _id của môn học từ component monhoc-item sang monhoc-table và đến monhoc.

Tại component monhoc-item, khởi tạo hàm handleDelete(id, _id)

```
handleDelete(id, _id) {
    this.props.onClickDelete(id, _id);
  }
```

và khai báo hàm trong constructor

```
    this.handleDelete = this.handleDelete.bind(this);
```

và gọi hàm trong phương thức onClick của nút Xóa

```
          <button className="btn btn-danger btn-
sm" onClick={() => this.handleDelete(item.id, item._id)}>
            Xóa
          </button
```

Tại component monhoc-table, thêm props onClickDelete cho component MonHocItem

```
<MonHocItem
          key={index}
          item={item}
          index={index}
          onClickEdit={this.props.onClickEdit}
          onClickDelete={this.props.onClickDelete}
/>
```

Tại component môn học, bổ sung props onClickDelete cho component TableMonHoc để có thể nhận được id của môn học cần xóa.

```
<TableMonHoc
          items={items}
          onClickEdit={this.handleEdit}
       onClickDelete={this.handleDelete}
/>
```

Thực hiện xóa dữ liệu

Trong component monhoc, khởi tạo hàm handleDelete để thực hiện cập nhật dữ liệu bằng cách gửi request với phương thức DELETE đến http://localhost:4000/monhoc/{id}

```
async handleDelete(id, _id) {
    if (window.confirm("Bạn muốn xóa môn học id: " + id + "?")) {
      await api
        .delete("/monhoc/" + _id)
        .then(response => {
          if (response.data == "fk") {
            alert("Vui lòng xóa dữ liệu bên bảng học phần trước");
          } else {
            alert("Đã xóa thành công!");
          }
        })
```

```
        .catch(err => console.log(err));
      await this.handleGet();
    }
  }
```

Đồng thời khai báo hàm vừa tạo trong constructor:

```
this.handleDelete = this.handleDelete.bind(this);
```

3.2.4.7 Thiết kế Private Route

Priavate Route là đường dẫn bắt buộc người dùng phải đăng nhập mới có thể truy cập được vào nội dung bên trong. Ý tưởng sẽ tạo một đối tượng bao gồm :

- State isAuthenticated để lưu lại trạng thái kiểm tra người dùng đã đăng nhập hay chưa.
- Hàm authenticate để set cho state trạng thái đã đăng nhập
- Hàm signout để set lại về trạng thái chưa đăng nhập.
- Hàm getLocalStorage_IsAuthenticated để kiểm tra giá trị của biến isAuthenticated được lưu trong local storage
- Hàm setLocalStorage_IsAuthenticated(value) để gán giá trị value tương ứng cho biến isAuthenticated được lưu trong local storage

```
exports.fakeAuth = {
  isAuthenticated: false,
  authenticate(cb) {
    this.isAuthenticated = true;
    setTimeout(cb, 100);
  },
  signout(cb) {
    this.isAuthenticated = false;
    setTimeout(cb, 100);
  },
  getLocalStorage_IsAuthenticated() {
    if (localStorage.getItem("isAuthenticated") === "true") {
      this.authenticate();
    } else {
      this.signout();
    }
  },
  setLocalStorage_IsAuthenticated(value) {
    localStorage.setItem("isAuthenticated", value);
  }
};
```

Một Private route cần đảm bảo các yêu cầu sau:

- Có các tính năng, các thuộc tính y hệt như component <Route />
- Render ra một component <Route /> và truyền tất cả các props được truyền vào cho component Route đó.
- Có khả năng xác thực người dùng đã đăng nhập hay chưa. Nếu họ đã đăng nhập thì render ra component được truyền vào còn không thì redirect họ đến đường dẫn /login.

Kết hợp các yêu cầu trên, xây dựng được component PrivateRoute.

```
fakeAuth.getLocalStorage_IsAuthenticated();
const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route
    {...rest}
    render={props => (fakeAuth.isAuthenticated === true
      ? <Component {...props} /> : <Redirect to="/login" />)
    }
  />
);
export default PrivateRoute;
```

Khai báo trong component App để có thể sử dụng Private route.

```
function App() {
  return (
    <Router>
      <div>
        <AuthHeader />
        <PrivateRoute exact path="/danhsach/monhoc" component={MonHoc} />
        <Route exact path="/login" component={Login} />
        <AuthFooter />
      </div>
    </Router>
  );
}
export default App;
```

3.2.4.8 Xây dựng chức năng đăng nhập

Tạo component login-form

Mô tả: gồm 1 form cho người dùng đăng nhập. Chi tiết mã lệnh nằm trong phần mục lục.

Kiểm tra thông tin admin

Khởi tạo hàm onSubmit để thực hiện kiểm tra có thông tin admin với username và password đã nhập bằng cách gửi request với phương thức POST đến http://localhost:4000/admin/login.

Nếu đăng nhập thành công, sẽ điều hướng đến trang chủ.

```
onSubmit() {
    const obj = {
      username: this.state.username,
      password: this.state.password
    };
    api.post("/admin/login", obj).then(res => {
      if (res.data === "admin not found") {
        alert("Tên đăng nhập hoặc mật khẩu sai!");
        this.setState({ username: "", password: "" });
      } else {
        alert("Đăng nhập thành công!");
        fakeAuth.setLocalStorage_IsAuthenticated(true);
        fakeAuth.authenticate(() => {
          this.setState(() => ({
            redirectToReferrer: true
          }));
        });
      }
    });
  }
```

3.3. Kiểm thử

Bảng 3.7 Bảng kiểm thử chức năng đăng nhập

STT	Mô tả dữ liệu kiểm thử	Kết quả mong đợi	Thành công/ Thất bại
1	Đăng nhập đúng tên đăng nhập và mật khẩu tài khoản admin	Hiển thị thông báo "Đăng nhập thành công!" và chuyển hướng đến trang chủ trang quản trị	Thành công
2	Đăng nhập sai tên đăng nhập hoặc mật khẩu tài khoản admin	Hiển thị thông báo "Tên đăng nhập hoặc mật khẩu sai!"	Thành công

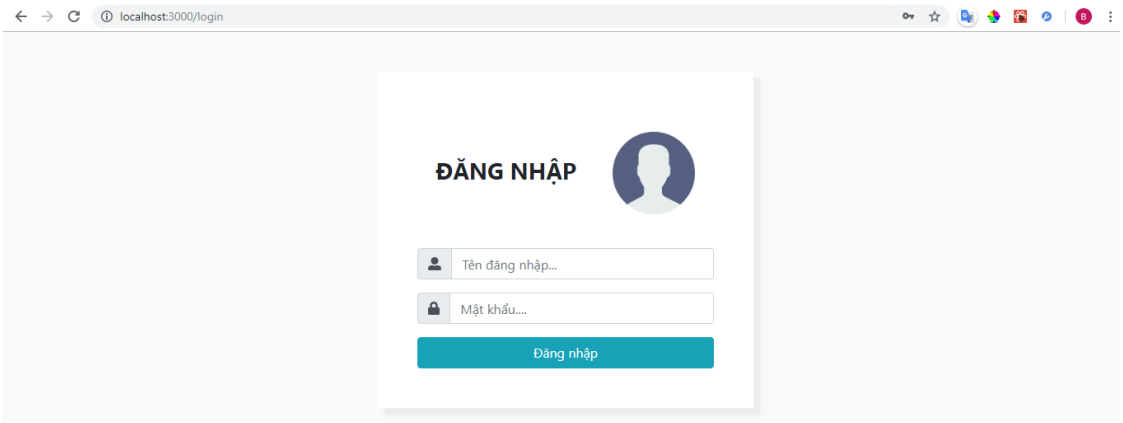
3	Đăng nhập đúng tên đăng nhập và mật khẩu tài khoản sinh viên	Hiển thị thông báo "Đăng nhập thành công!" và chuyển hướng đến trang đăng ký học phần	Thành công
4	Đăng nhập sai tên đăng nhập hoặc mật khẩu tài khoản sinh viên	Hiển thị thông báo "Tên đăng nhập hoặc mật khẩu sai!"	Thành công
5	Truy cập vào các trang nội dung khi chưa đăng nhập	Điều hướng sang trang đăng nhập	Thành công

Bảng 3.8 Bảng kiểm thử chức năng thêm, cập nhật, xóa dữ liệu

STT	**Mô tả dữ liệu kiểm thử**	**Kết quả mong đợi**	**Thành công/ Thất bại**
1	Tạo dữ liệu với đầy đủ thông tin và đúng định dạng	Hiển thị thông báo "Đã thêm thành công!" và chuyển hướng đến trang Danh sách của dữ liệu	Thành công
2	Tạo dữ liệu thiếu trường ID	Hiển thị thông báo "Chưa nhập ID!"	Thành công
3	Cập nhật thông tin dữ liệu đúng định dạng	Hiển thị thông báo "Đã cập nhật thành công!" và chuyển hướng đến trang Danh sách của dữ liệu	Thành công
4	Xóa dữ liệu không có khóa ngoại	Hiển thị xác nhận xóa. Nếu xác nhận đồng ý xóa, báo đã xóa thành công.	Thành công
5	Xóa dữ liệu không có khóa ngoại	Hiển thị xác nhận xóa. Nếu xác nhận đồng ý xóa, hiển thị thông báo "Vui lòng xóa dữ liệu ở bảng con trước!".	Thành công

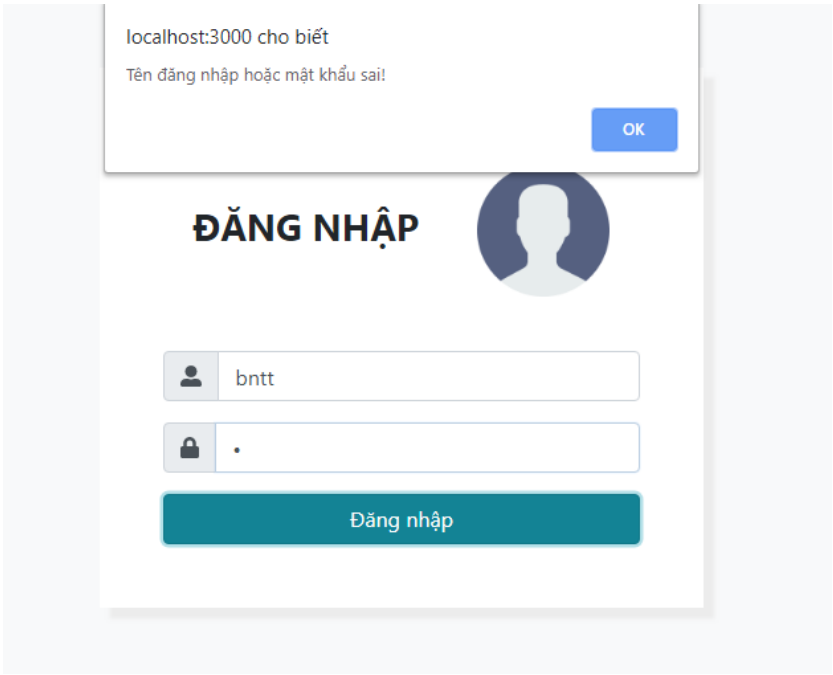
3.4. Kết quả đạt được

3.4.1.1 Chức năng đăng nhập cho admin



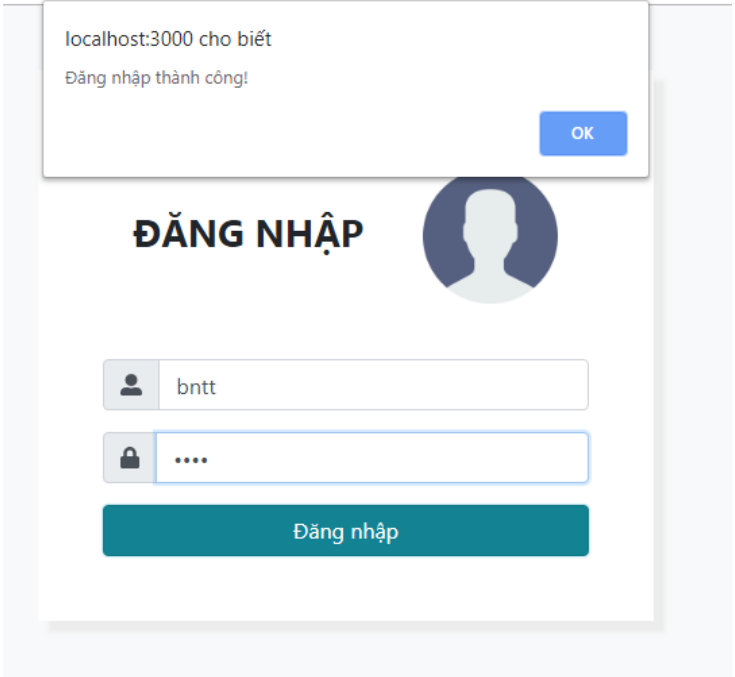
Hình 3.3 Màn hình đăng nhập

Khi người dùng đăng nhập vào hệ thống, nếu chưa từng đăng nhập, người dùng sẽ được điều hướng sang trang Đăng nhập để thực hiện việc xác thực thông tin.



Hình 3.4 Màn hình khi đăng nhập sai tên hoặc mật khẩu admin

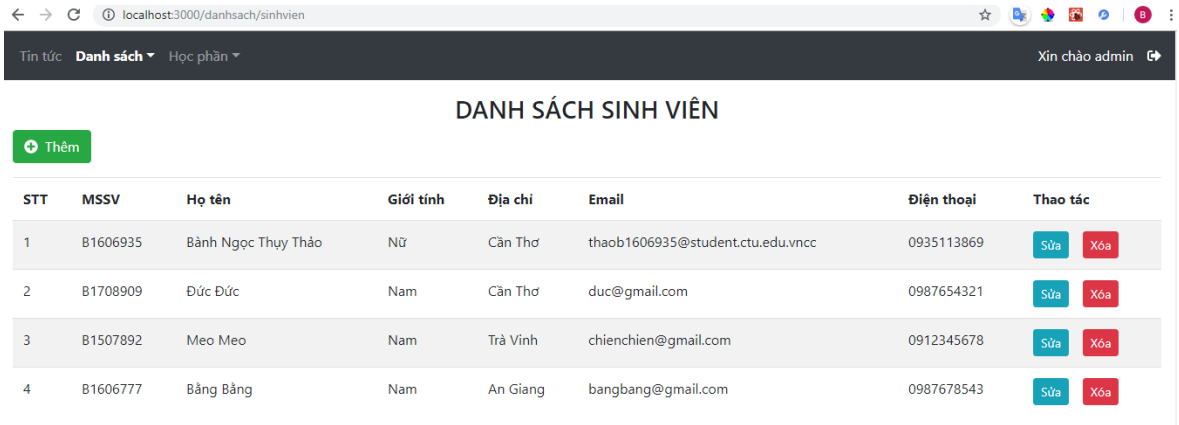
Nếu người dùng nhập sai tên đăng nhập hoặc mật khẩu tài khoản admin, sẽ có thông báo "Tên đăng nhập hoặc mật khẩu sai". Trường hợp dùng tài khoản sinh viên để đăng nhập cũng sẽ có thông báo tương tự.



Hình 3.5 Màn hình khi đăng nhập đúng tên và mật khẩu admin

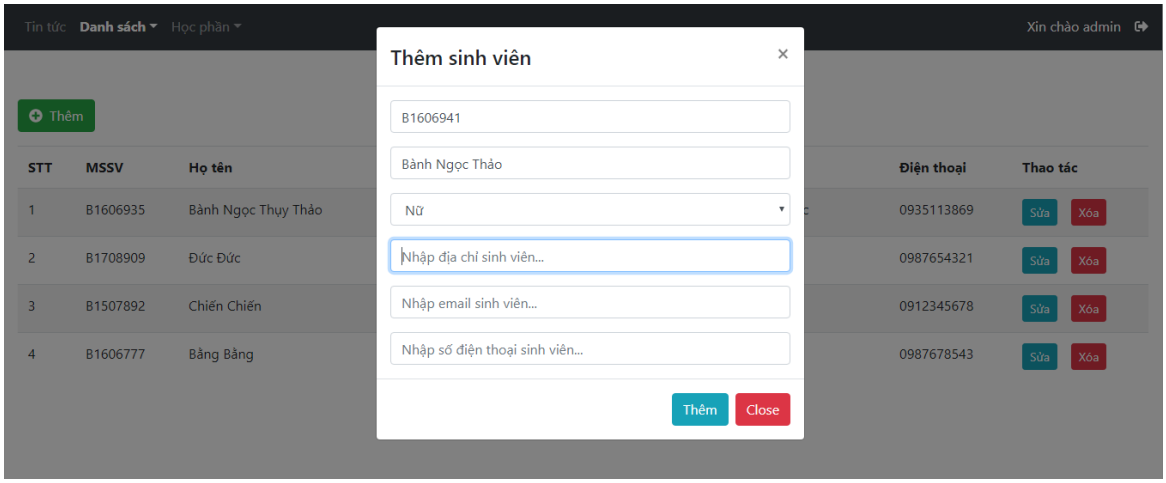
Khi người dùng đăng nhập đúng tên và mật khẩu tài khoản admin, sẽ có thông báo hiển thị "Đăng nhập thành công" và điều hướng đến trang chủ hệ thống. Thông tin đăng nhập sẽ được lưu lại.

3.4.1.2 Trang thông tin sinh viên



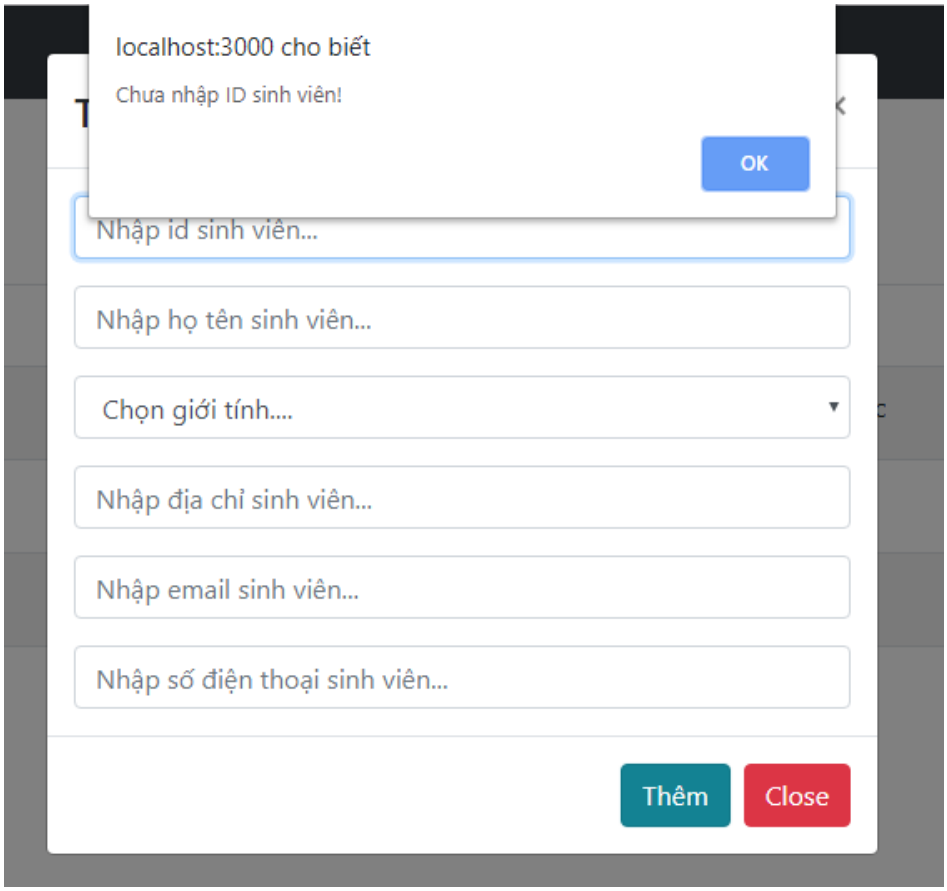
Hình 3.6 Màn hình trang hiển thị danh sách sinh viên

Trang thông tin sinh viên sẽ hiển thị danh sách tất cả sinh viên có trong cơ sở dữ liệu. Đồng thời cũng cho phép admin thêm một sinh viên mới, sửa các thông tin hay xóa sinh viên.

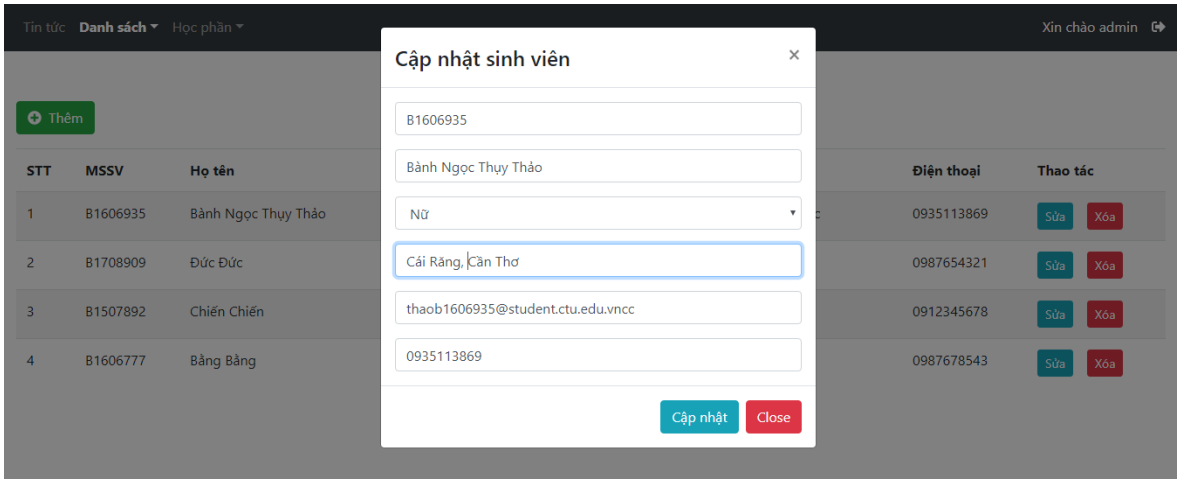


Hình 3.7 Màn hình thêm thông tin sinh viên

Khi thêm một sinh viên mới, admin bắt buộc nhập thông tin trường Sinh viên ID. Nếu để rỗng, sẽ có thông báo "Chưa nhập ID sinh viên!".

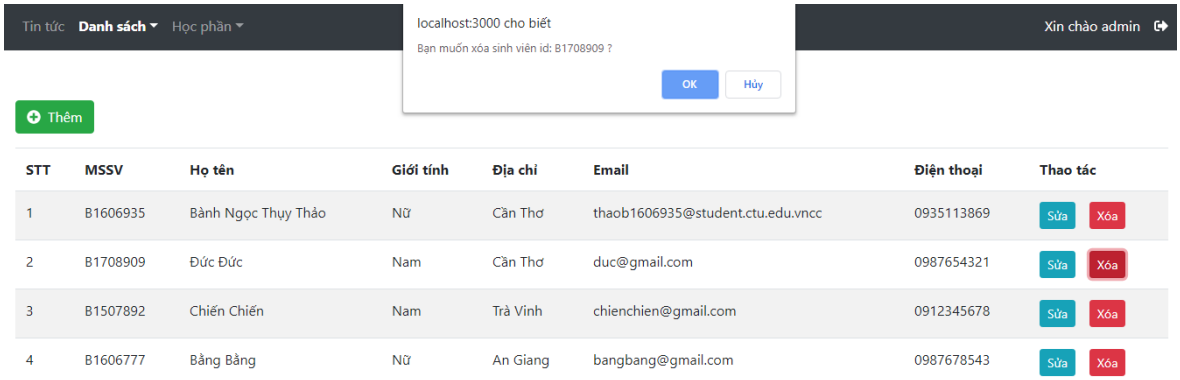


Hình 3.8 Thông báo khi chưa nhập ID sinh viên



Hình 3.9 Màn hình cập nhật thông tin sinh viên

Khi admin muốn cập nhật thông tin sinh viên, sẽ nhấn vào nút "Sửa" trên trường thông tin tương ứng. Hệ thống không cho phép sửa đổi trường ID sinh viên. Ngoài ra, những trường thông tin khác được phép cập nhật hoặc để rỗng.



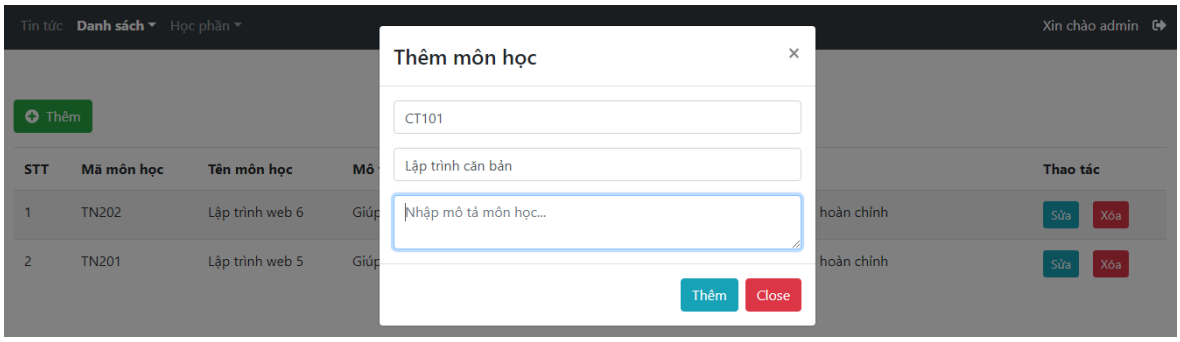
Hình 3.10 Màn hình xóa sinh viên

Khi người dùng muốn xóa sinh viên, sẽ nhấn vào nút "Xóa" trên trường dữ liệu tương ứng. Khi đó sẽ có thông báo xác nhận. Nếu người dùng bấm "OK", sẽ thực hiện xóa sinh viên.

3.4.1.3 Trang môn học

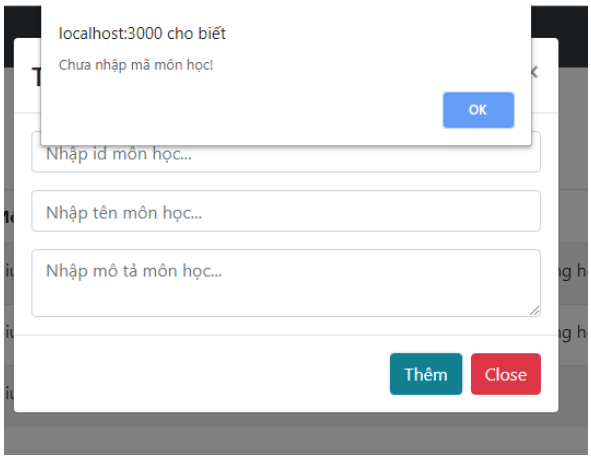
Hình 3.11 Màn hình thông tin môn học

Trang thông tin môn học sẽ hiển thị danh sách tất cả môn học có trong cơ sở dữ liệu. Đồng thời cũng cho phép admin thêm một môn học mới, sửa các thông tin hay xóa môn học.

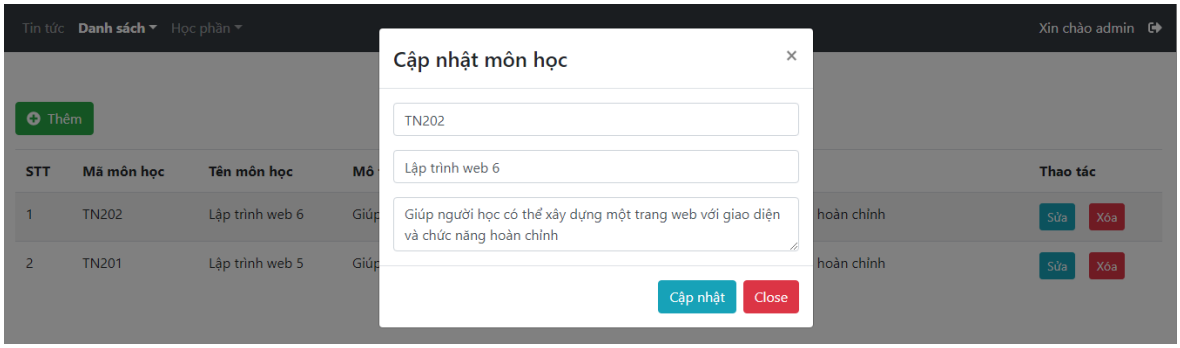


Hình 3.12 Màn hình thêm môn học

Khi thêm một môn học mới, admin bắt buộc nhập thông tin trường môn học ID. Nếu để rỗng, sẽ có thông báo “Chưa nhập ID môn học!”.

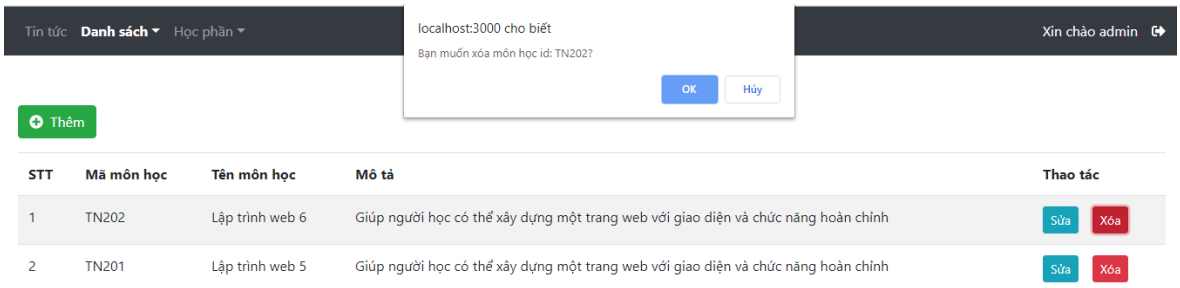


Hình 3.13 Thông báo khi chưa nhập mã môn học.



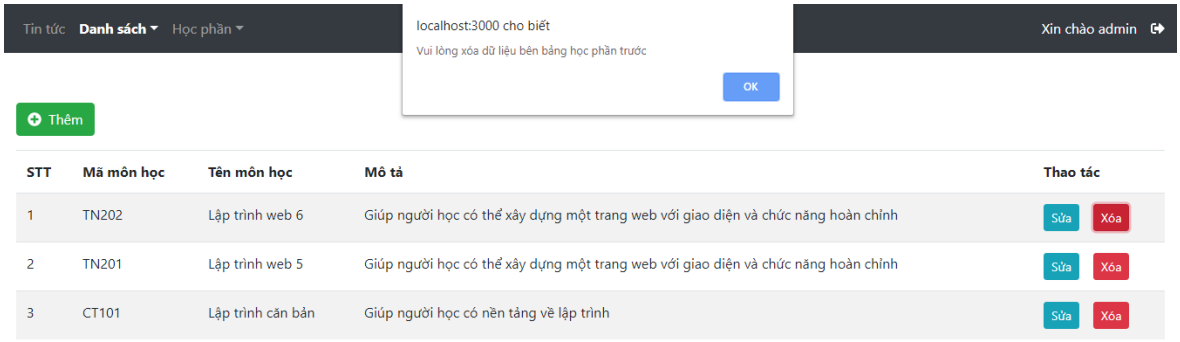
Hình 3.14 Màn hình cập nhật môn học

Khi admin muốn cập nhật thông tin môn học, sẽ nhấn vào nút "Sửa" trên trường thông tin tương ứng. Hệ thống không cho phép sửa đổi trường ID môn học. Ngoài ra, những trường thông tin khác được phép cập nhật hoặc để rỗng.



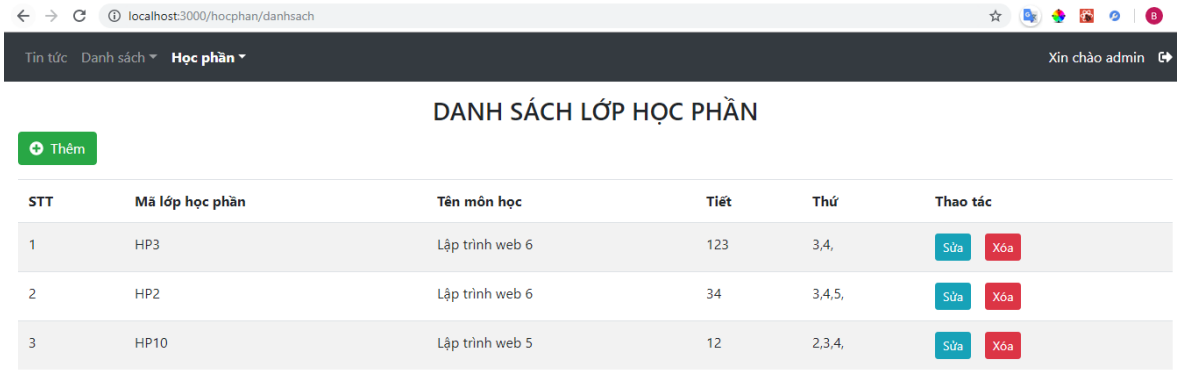
Hình 3.15 Màn hình xóa môn học

Khi người dùng muốn xóa sinh viên, sẽ nhấn vào nút "Xóa" trên trường dữ liệu tương ứng. Khi đó sẽ có thông báo xác nhận. Nếu người dùng bấm "OK", sẽ thực hiện xóa sinh viên. Tuy nhiên, nếu dữ liệu môn học có khóa ngoại, tức là tồn tại lớp học phần của môn học đó, sẽ không cho phép xóa môn học này. Khi đó sẽ có thông báo "Vui lòng xóa dữ liệu bên bảng học phần trước!". Ngược lại sẽ có thông báo "Đã xóa thành công!".



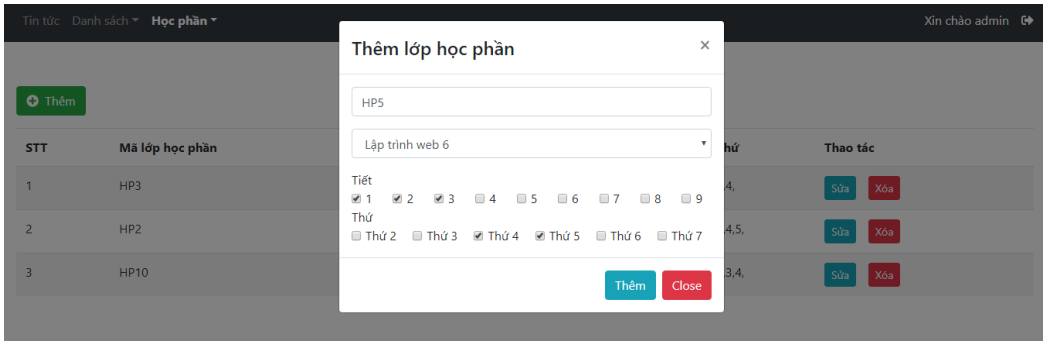
Hình 3.16 Màn hình khi xóa môn học có khóa ngoại

3.4.1.4 Trang lớp học phần



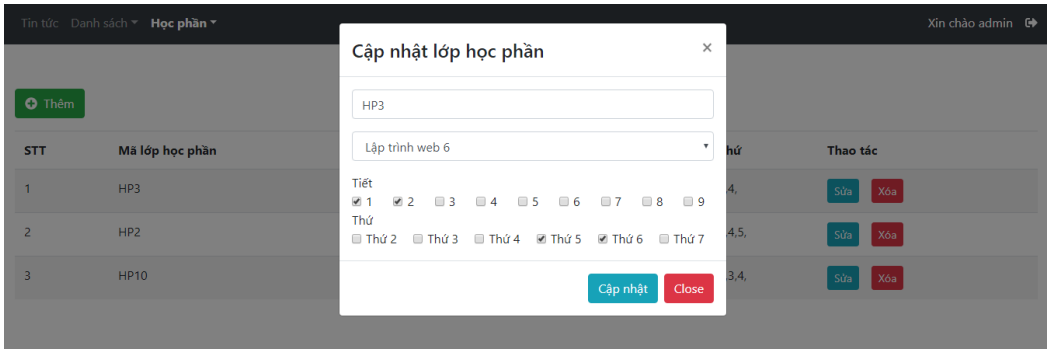
Hình 3.17 Màn hình danh sách lớp học phần

Trang thông tin lớp học phần sẽ hiển thị danh sách tất cả lớp học phần có trong cơ sở dữ liệu. Đồng thời cũng cho phép admin thêm một lớp học phần mới, sửa các thông tin hay xóa lớp học phần.



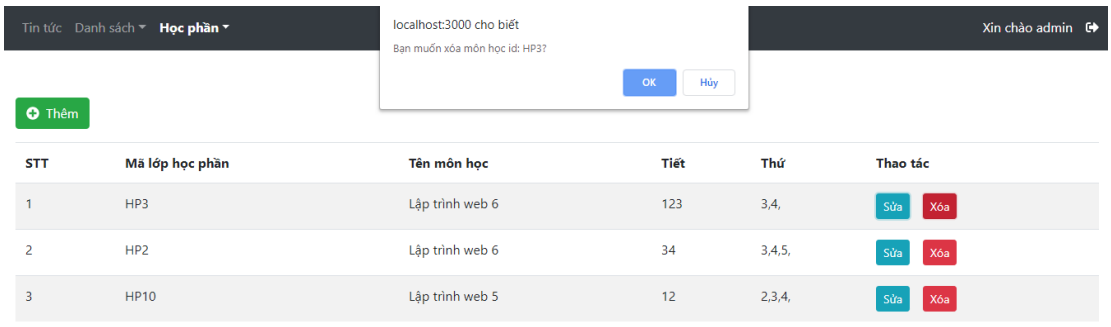
Hình 3.18 Màn hình thêm lớp học phần

Khi thêm một lớp học phần mới, admin bắt buộc nhập thông tin trường mã lớp học phần. Nếu để rỗng, sẽ có thông báo "Chưa nhập ID lớp học phần!".



Hình 3.19 Màn hình cập nhật lớp học phần

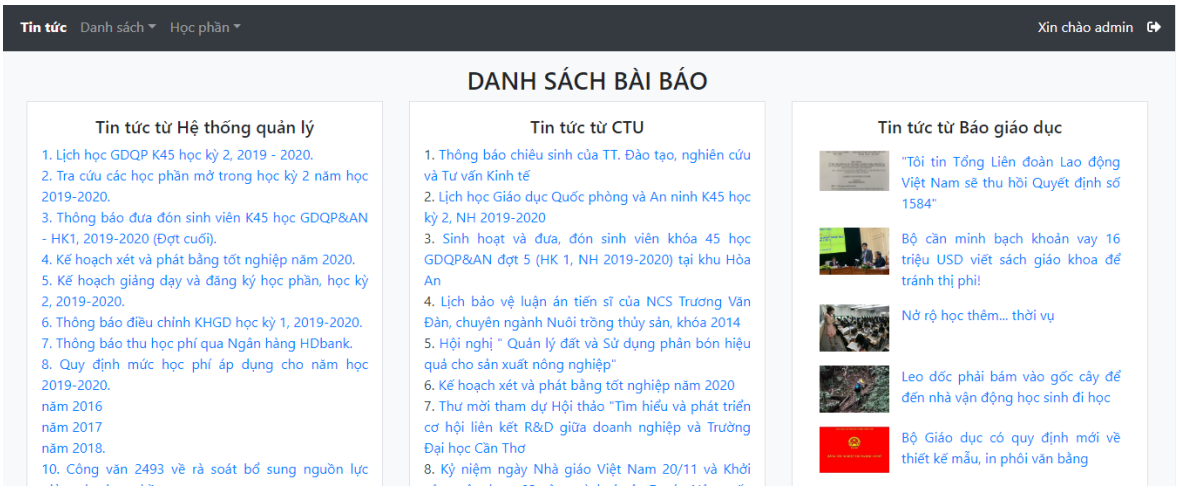
Khi admin muốn cập nhật thông tin lớp học phần, sẽ nhấn vào nút "Sửa" trên trường thông tin tương ứng. Hệ thống không cho phép sửa đổi trường ID lớp học phần. Ngoài ra, những trường thông tin khác được phép cập nhật hoặc để rỗng.



Hình 3.20 Màn hình xóa lớp học phần

Khi người dùng muốn xóa lớp học phần, sẽ nhấn vào nút "Xóa" trên trường dữ liệu tương ứng. Khi đó sẽ có thông báo xác nhận. Nếu người dùng bấm "OK", sẽ thực hiện xóa lớp học phần.

3.4.1.5 Trang tin tức



Hình 3.21 Màn hình trang tin tức

Trang tin tức sẽ hiển thị những thông tin từ các trang htql.ctu.edu.vn, ctu.edu.vn và giaoduc.net.vn. Người dùng có thể nhấn trực tiếp vào tựa báo tương ứng để xem nội dung bài báo.

CHƯƠNG 4: KẾT LUẬN, HƯỚNG PHÁT TRIỂN

Đề tài "Tìm hiểu và xây dựng RESTFul API web service theo mô hình MERN Stack" đã trình bày được những kiến thức về web service, RESTFul API, MongoDB, Express, React, NodeJS và những thư viện, framework liên quan. Đề tài cũng trình bày được các bước xây dựng một web service theo chuẩn RESTFul từ bước thiết kế API, xây dựng server và thiết kế giao diện, cũng như thực hiện được những chức năng cơ bản của một trang web quản lý thông tin.

Đề tài là nền tảng cơ bản để xây dựng nên một hệ thống quản lý thông tin với đầy đủ chức năng hơn và có thể áp dụng vào thực tế. Ngoài ra, đây cũng là tiền đề để người lập trình tìm hiểu sâu hơn về các công nghệ web khác, giúp hoàn thiện bản thân mình.

TÀI LIỆU THAM KHẢO

[1] Bootstrap, https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework), truy cập tháng 10/2019

[2] Cory Gackenheimer, "Introduction to React", xuất bản 2015

[3] Cơ bản về Router trong ReactJs, https://viblo.asia/p/co-ban-ve-router-trong-reactjs-07LKXzAElV4, truy cập tháng 10/2019

[4] CSS, https://vi.wikipedia.org/wiki/CSS, truy cập tháng 10/2019

[5] Dịch vụ web, https://vi.wikipedia.org/wiki/D%E1%BB%8Bch_v%E1%BB%A5_web, truy cập tháng 10/2019

[6] Express.js, https://en.wikipedia.org/wiki/Express.js, truy cập tháng 10/2019

[7] Giới thiệu về Mongoose cho MongoDB và NodeJS, https://code.tutsplus.com/vi/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527, truy cập tháng 10/2019

[8] HTML, https://vi.wikipedia.org/wiki/HTML, truy cập tháng 10/2019

[9] JavaScript, https://vi.wikipedia.org/wiki/JavaScript, truy cập tháng 10/2019

[10] Kristen Dyrr, "The Complete Beginner's Guide to React", xuất bản 2018

[11] Manuel Kiessling, "The node craftsman book", xuất bản 2017

[12] MongoDB, https://en.wikipedia.org/wiki/MongoDB, truy cập tháng 10/2019

[13] Mô hình MVC là gì? https://khoanguyen.me/tim-hieu-mo-hinh-mvc-la-gi/, truy cập tháng 10/2019

[14] NodeJS, https://freetuts.net/nodejs-la-gi-584.html, truy cập tháng 10/2019

[15] React docs, https://reactjs.org/, truy cập tháng 09/2019

[16] React training, https://reacttraining.com/react-router/web/guides/quick-start, truy cập tháng 11/2019

[17] RESTful API là gì? Cách thiết kế RESTful API, https://topdev.vn/blog/restful-api-la-gi/, truy cập tháng 10/2019

[18] Tạo app CRUD đơn giản với ReactJs, NodeJs, MongoDB, https://viblo.asia/p/tao-app-crud-don-gian-voi-reactjs-nodejs-mongodb-ORNZqBnMl0n, truy cập tháng 11/2019

PHỤ LỤC

PL1. Mã lệnh chi tiết các hàm trong monhoc-controller.js

```
const MonHoc = require("../models/monhoc-model");
const DSHP = require("../models/danhsachhocphan-model");
//Lấy danh sách dữ liệu
exports.getListMonHoc = (req, res) => {
  MonHoc.find(function(err, mh) {
    if (err) {
      console.log(err);
    } else {
      res.json(mh);
    }
  });
};

// Lấy dữ liệu theo khóa chính
exports.getMonHoc = (req, res) => {
  MonHoc.findById(req.params.id, function(err, mh) {
    if (err) {
      console.log(err);
    } else {
      res.json(mh);
    }
  });
};

//Thêm mới dữ liệu
exports.addMonHoc = (req, res) => {
  let mh = new MonHoc(req.body);
  mh.save()
    .then(mh => {
      res.status(200).json({ mh: "subject in added successfully" });
    })
    .catch(err => {
      res.status(400).send("unable to save to database");
    });
};

//Xóa dữ liệu
exports.deleteMonHoc = (req, res) => {
  DSHP.findOne({ idMonHoc: req.params.id }, function(err, dshp) {
    if (err) res.json(err);
    else {
      if (!dshp) {
```

```
        MonHoc.findByIdAndRemove(req.params.id, function(err, mh) {
          if (err) res.json(err);
          else res.json("Successfully removed");
        });
      } else {
        res.json("fk");
      }
    }
  });
};

//Cập nhật dữ liệu
exports.updateMonHoc = (req, res) => {
  MonHoc.findById(req.params.id, function(err, mh) {
    if (!mh) res.status(404).send("data is not found");
    else {
      console.log(mh);
      mh.id = req.body.id;
      mh.name = req.body.name;
      mh.description = req.body.description;
      mh.save()
        .then(mh => {
          res.json("Update complete");
        })
        .catch(err => {
          res.status(400).send("unable to update the database");
        });
    }
  });
};
```

PL2. Mã lệnh chi tiết component AddForm

```
class AddForm extends Component {
  constructor(props) {
    super(props);
    this.state = { isShowAddForm: false };
    this.handleAddForm = this.handleAddForm.bind(this);
  }
  //---------Functions
  handleAddForm() {
    this.setState({ isShowAddForm: !this.state.isShowAddForm });
    console.log(this.state.isShowAddForm);
  }

  render() {
```

```
    var isShowAddForm = this.state.isShowAddForm;
    var elAddForm = null;
    if (isShowAddForm) {
      elAddForm = <MonHocAddForm isShowAddForm={this.state.isShowAddForm} on
ClickClose={this.handleAddForm} />;
    }
    return (
      <div>
        <button className="btn btn-success mb-
3" onClick={this.handleAddForm}>
          <FontAwesomeIcon icon={faPlusCircle} className="mr-2" />
          Thêm
        </button>
        {elAddForm}
      </div>
    );
  }
}
export default AddForm;
```

PL3. Mã lệnh chi tiết component MonHocAddForm

```
class MonHocAddForm extends Component {
  constructor(props) {
    super(props);
    this.state = { id: "", name: "", description: "" };
    this.onChangeDescription = this.onChangeDescription.bind(this);
    this.onChangeName = this.onChangeName.bind(this);
    this.onChangeId = this.onChangeId.bind(this);
    this.close = this.close.bind(this);
  }
  //-----onChangeId
  onChangeId(e) {
    this.setState({ id: e.target.value });
  }
  //-----onChangeName
  onChangeName(e) {
    this.setState({ name: e.target.value });
  }
  //-----onChangeDescription
  onChangeDescription(e) {
    this.setState({ description: e.target.value });
  }
  close() {
    this.props.onClickClose();
  }
```

```
  render() {
    return (
      <form className="d-flex">
        <Modal show={this.props.isShowAddForm} onHide={this.close}>
          <Modal.Header closeButton>
            <Modal.Title>Thêm môn học</Modal.Title>
          </Modal.Header>
          <Modal.Body>
            <input
              className="form-control"
              placeholder="Nhập id môn học..."
              required
              value={this.state.id}
              onChange={this.onChangeId}
            />
            <input
              className="form-control mt-3"
              placeholder="Nhập tên môn học..."
              required
              value={this.state.name}
              onChange={this.onChangeName}
            />
            <textarea
              className="form-control mt-3"
              placeholder="Nhập mô tả môn học..."
              required
              value={this.state.description}
              onChange={this.onChangeDescription}
            ></textarea>
          </Modal.Body>
          <Modal.Footer>
            <input type="submit" value="Thêm" className="btn btn-
info" onClick={this.onSubmit} />
            <Button onClick={this.close} className="btn-danger">
              Close
            </Button>
          </Modal.Footer>
        </Modal>
      </form>
    );
  }
}
export default MonHocAddForm;
```

PL4. Mã lệnh chi tiết component MonHocEditForm

```
class MonHocEditForm extends Component {
constructor(props) {
    super(props);
    this.state = { id: "", name: "", description: "", _id: "" };
    this.onChangeId = this.onChangeId.bind(this);
    this.onChangeName = this.onChangeName.bind(this);
    this.onChangeDescription = this.onChangeDescription.bind(this);
    this.close = this.close.bind(this);
  }
  //---------Functions
  //-----onChangeId
  onChangeId(e) {
    this.setState({ id: e.target.value });
  }
  //-----onChangeName
  onChangeName(e) {
    this.setState({ name: e.target.value });
  }
  //-----onChangeGender
  onChangeDescription(e) {
    this.setState({ description: e.target.value });
  }
  close() {
    this.props.onClickClose();
  }
  render() {
    return (
      <form className="d-flex">
        <Modal show={true} onHide={this.close}>
          <Modal.Header closeButton>
            <Modal.Title>Cập nhật môn học</Modal.Title>
          </Modal.Header>

          <Modal.Body>
            <input
              className="form-control"
              placeholder="Nhập id môn học..."
              required
              value={this.state.id}
              onChange={this.onChangeId}
            />
            <input
              className="form-control mt-3"
              placeholder="Nhập tên môn học..."
```

```
              required
              value={this.state.name}
              onChange={this.onChangeName}
            />
            <textarea
              className="form-control mt-3"
              placeholder="Nhập mô tả môn học..."
              required
              value={this.state.description}
              onChange={this.onChangeDescription}
            ></textarea>
          </Modal.Body>
          <Modal.Footer>
            <input type="submit" value="Cập nhật" className="btn btn-
info" onClick={this.onSubmit} />
            <Button onClick={this.close} className="btn-danger">
              Close
            </Button>
          </Modal.Footer>
        </Modal>
      </form>
    );
  }
}

export default MonHocEditForm;
```

PL5. Mã lệnh chi tiết component MonHoc

```
class MonHoc extends Component {
  constructor(props) {
    super(props);
    this.state = {
      items: [],
      isShowEditForm: false,
      itemUpdated: null,
      updated: false
    };
    this.handleEdit = this.handleEdit.bind(this);
    this.handleClose = this.handleClose.bind(this);
  }

  handleEdit() {
    this.setState({ isShowEditForm: true});
  }
  handleClose() {
    this.setState({ isShowEditForm: false });
```

```
  }

  render() {
    var items = this.state.items;
    var isShowEditForm = this.state.isShowEditForm;
    var elShowEditForm = "";
    if (isShowEditForm) {
      elShowEditForm = (
        <MonHocEditForm
          onClickClose={this.handleClose}
          isShowEditForm={this.state.isShowEditForm}
        />
      );
    }
    return (
      <div
        className="bg-white table-striped p-3"
        style={{ minHeight: "100vh" }}
      >
        <h3 className="text-center">DANH SÁCH MÔN HỌC</h3>
        <AddForm />
        {elShowEditForm}
        <TableMonHoc
          items={items}
        />
      </div>
    );
  }
}
```

PL6. Mã lệnh chi tiết component TableMonHoc

```
class TableMonHoc extends Component {
  render() {
    var items = this.props.items;
    var elItem = items.map((item, index) => {
      return (
        <MonHocItem
          key={index}
          item={item}
          index={index}
        />
      );
    });
    return (
      <table className="table">
```

```
        <thead>
          <tr>
            <td>STT</td>
            <td>Mã môn học</td>
            <td>Tên môn học</td>
            <td>Mô tả</td>
            <td>Thao tác</td>
          </tr>
        </thead>
        <tbody>{elItem}</tbody>
      </table>
    );
  }
}
export default TableMonHoc;
```

PL7. Mã lệnh chi tiết component MonHocItem

```
class MonHocItem extends Component {
  constructor(props) {
    super(props);
  }
  render() {
    var { item } = this.props;
    var { index } = this.props;
    return (
      <tr>
        <td>{index + 1}</td>
        <td>{item.id}</td>
        <td>{item.name}</td>
        <td>{item.description}</td>
        <td>
          <button className="btn btn-info mr-3 btn-sm">
            Sửa
          </button>
          <button className="btn btn-danger btn-sm">
            Xóa
          </button>
        </td>
      </tr>
    );
  }
}
export default MonHocItem;
```

PL8. Mã lệnh chi tiết component LoginForm

```
class LoginForm extends Component {
  constructor(props) {
    super(props);
    this.state = { username: "", password: "", redirectToReferrer: false };
    this.onChangePassword = this.onChangePassword.bind(this);
    this.onChangeUsername = this.onChangeUsername.bind(this);
  }

  //onChangeUsername
  onChangeUsername(e) {
    this.setState({ username: e.target.value });
  }
  //onChangePassword
  onChangePassword(e) {
    this.setState({ password: e.target.value });
  }
  render() {
    const { redirectToReferrer } = this.state;
    if (redirectToReferrer === true) {
      return <Redirect to="/" />;
    }

    return (
      <form>
        <div className="my-3">
          <div className="input-group mb-2">
            <div className="input-group-prepend">
              <div className="input-group-text">
                <FontAwesomeIcon icon={faUserAlt} />
              </div>
            </div>
            <input
              type="text"
              className="form-control"
              id="txtUsername"
              placeholder="Tên đăng nhập..."
              value={this.state.username}
              onChange={this.onChangeUsername}
              required
            />
          </div>
        </div>
        <div className="my-3">
          <div className="input-group mb-2">
```

```
            <div className="input-group-prepend">
              <div className="input-group-text">
                <FontAwesomeIcon icon={faLock} />
              </div>
            </div>
            <input
              type="password"
              className="form-control"
              id="txtPasssword"
              placeholder="Mật khẩu...."
              value={this.state.password}
              onChange={this.onChangePassword}
              required
            />
          </div>
        </div>
        <input type="button" value="Đăng nhập" className="btn btn-block btn-
info" onClick={this.onSubmit} />
      </form>
    );
  }
}

export default LoginForm;
```