**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**BÁO CÁO BÀI TẬP LỚN**

**MÔN LẬP TRÌNH CƠ SỞ DỮ LIỆU**

**ĐỀ TÀI: TRANG WEB CHIA SẺ TÀI LIỆU HỌC TẬP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Giảng viên:**

**TP. HỒ CHÍ MINH, 2021**

**MỤC LỤC**

[Chương 1. React **Error! Bookmark not defined.**](#_Toc94482527)

[1.1. Giới thiệu về React 22](#_Toc94482528)

[1.2. Ưu điểm của React 22](#_Toc94482529)

[1.2.1 Khắc phục nhược điểm của framework 22](#_Toc94482530)

[1.2.2 Ưu điểm khiến React trở nên phổ biến 22](#_Toc94482531)

[1.3. Một số khái niệm cơ bản về React 22](#_Toc94482532)

[1.3.1 JSX 22](#_Toc94482533)

[1.3.2 Component 22](#_Toc94482534)

[1.3.3 Props 23](#_Toc94482535)

[1.3.4 State 23](#_Toc94482536)

[1.3.5 Vòng đời 23](#_Toc94482537)

[1.3.6 React Router 25](#_Toc94482538)

[1.3.7 Hooks Effect 25](#_Toc94482539)

[1.3.8 React Axious 25](#_Toc94482540)

[1.3.9 React Redux 25](#_Toc94482541)

[1.3.8.3 Store 25](#_Toc94482542)

[1.3.8.4 Cơ chế hoạt động 25](#_Toc94482543)

# WEBSITE CHIA SẺ TÀI LIỆU HỌC TẬP

Đây là chương em sẽ trình bày tổng quan về các lược đồ phân tích và thiết kế của hệ thống. Đồng thời, em sẽ mô tả chi tiết hoạt động của từng chức năng.

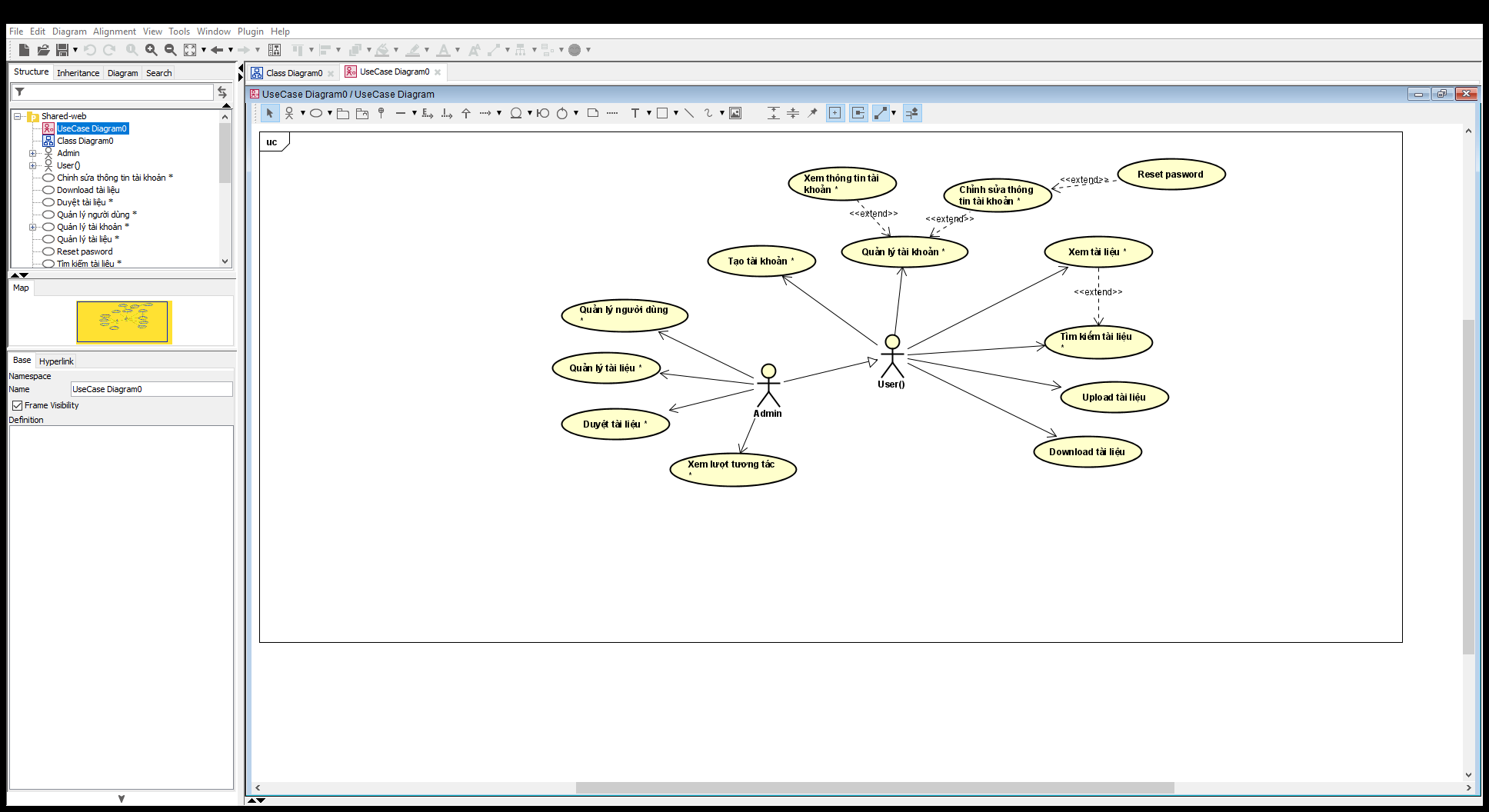
## Giới thiệu hệ thống

Website chia sẻ tài liệu học tập là trang web cho phép người dùng chia sẻ các loại tài liệu dưới dạng pdf, word…, những ai đăng kí tài khoản đều có thể tải tài liệu được chia sẻ về để học tập, đây là nơi cung cấp kiến thức miễn phí cho cộng đồng uy tín chất lượng. Các chức năng của hệ thống gồm đăng nhập, đăng kí, upload tài liệu, download tài liệu, tìm kiếm tài liệu theo môn học.

… . Hơn thế nữa, hệ thống còn cung cấp trang quản trị admin giúp người quản trị quản lý tài khoản người dùng, quản lý duyệt tài liệu trước khi được upload lên trang, giúp công việc quản lý hiệu quả hơn, uy tín hơn.

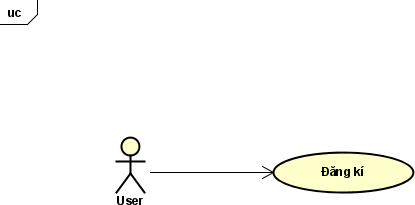
## Phân tích thiết kế hệ thống

### Lược đồ Use Case tổng quát

****

Hình 1.1: Lược đồ Use Case tổng quát

### Use Case đăng kí



Hình 1.2: Lược đồ Use Case đăng kí

**Đặc tả Use Case đăng kí**

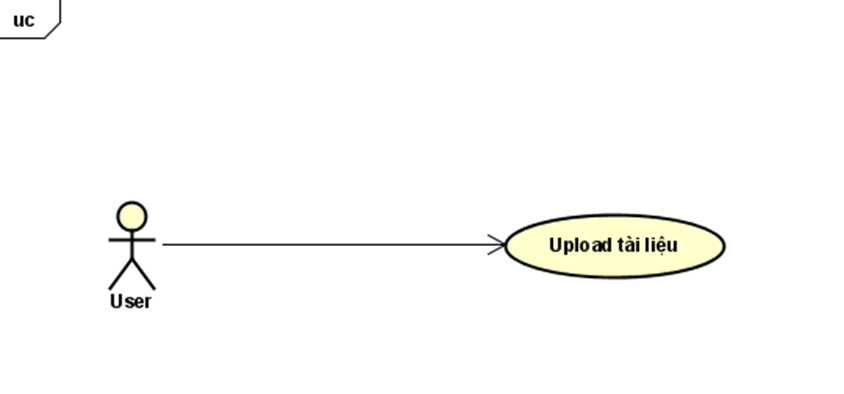
|  |  |
| --- | --- |
| **Use Case ID** | 1 |
| **Tên Use Case** | Đăng kí. |
| **Mô tả Use Case** | Use Case này cho phép người dùng tạo tài khoản để truy cập và sử dụng dịch vụ của hệ thống. |
| **Actor chính** | Người dùng (User). |
| **Actor phụ** | Không có. |
| **Tiền điều kiện** | User nhập đúng các thông tin hệ thống yêu cầu.  Có kết nối internet. |
| **Hậu điều kiện** | Hệ thống hiển thị giao diện thông báo đến người dùng biết họ đã đăng kí thành công. |
| **Luồng hoạt động** | 1. Người dùng vào hệ thống và chọn chức năng đăng kí. 2. Hệ thống yêu cầu người dùng nhập các thông tin cần thiết. 3. Người dùng nhập thông tin theo yêu cầu hệ thống. 4. Hệ thống kiểm tra thông tin. 5. Hệ thống hiển thị giao diện đăng kí thành công. |
| **Luồng thay thế** | Không có. |
| **Luồng ngoại lệ** | Ở bước 4 nếu hệ thông kiểm tra thông tin có chưa phù hợp yêu cầu và hiển thị thông báo yêu cầu nhập lại thông tin.   * Người dùng nhập lại thông tin như bước 3. * Người dùng chọn hủy đăng kí -> Use Case dừng lại. |
| Bảng 1.1: Đặc tả Use Case “Đăng kí” Use Case đăng nhập Diagram  Description automatically generated  Hình 1.3: Lược đồ Use Case đăng nhập | |

**Đặc tả Use Case đăng nhập**

|  |  |
| --- | --- |
| **Use Case ID** | 2 |
| **Tên Use Case** | Đăng nhập. |
| **Mô tả Use Case** | Use Case này cho phép người dùng đăng nhập và sử dụng các chức năng của hệ thống. |
| **Actor chính** | Người dùng (User). |
| **Actor phụ** | Không có. |
| **Tiền điều kiện** | User đã tạo tài khoản.  Có kết nối internet. |
| **Hậu điều kiện** | Hiển thị giao diện cho biết người dùng đăng nhập thành công. |
| **Luồng hoạt động** | 1. Người dùng truy cập hệ thống và chọn chức năng đăng nhập. 2. Hệ thống hiển thị giao diện đăng nhập. 3. Người dùng nhập thông tin tài khoản và mật khẩu. 4. Hệ thống kiểm tra thông tin tài khoản. 5. Hệ thống hiển thị thông báo đăng nhập thành công và chuyển hướng đến trang chủ của hệ thống. |
| **Luồng thay thế** | Không có. |
| **Luồng ngoại lệ** | Ở bước 4 nếu hệ thống kiểm tra thông  tin có chưa phù hợp và hiển thị thông báo sai tên đăng nhập hoặc mật khẩu, yêu cầu  nhập lại.   * Người dùng nhập lại tài khoản mật khẩu giống bước 3. |

Bảng 1.2: Đặc tả Use Case “Đăng nhập”

### Use Case upload tài liệu



Hình 1.4: Lược đồ Use Case upload tài liệu

**Đặc tả Use Case upload tài liệu**

|  |  |
| --- | --- |
| **Use Case ID** | 3 |
| **Tên Use Case** | Upload tài liệu |
| **Mô tả Use Case** | Use Case này cho phép người dùng upload tài liệu muốn chia sẻ.  Có thể chỉnh sửa tài liệu upload. |
| **Actor chính** | Người dùng (User). |
| **Actor phụ** | Không có. |
| **Tiền điều kiện** | User đã đăng nhập |
| **Hậu điều kiện** | Hiển thị thông tin người đăng tài liệu sau khi tài liệu được upload. |
| **Luồng hoạt động** | 1. Người dùng nhấn mục đăng tài liệu. 2. Hệ thống hiển thị giao diện để tải file tài liệu lên. 3. Trong quá trình tài liệu tải lên người dung tiến hành chọn chủ đề cho tài liệu và mô tả ngắn về tài liệu. 4. Sau khi điền đẩy đủ thông tin và tài liệu tải lên hoàn tất tiến hành đăng tài liệu. |
| **Luồng thay thế** | Không có. |
| **Luồng ngoại lệ** | Ở bước 3, nếu như thông tin mô tả và loại tài liệu không được chọn sẽ không cho phép đăng tài liệu. |

Bảng 1.3: Đặc tả Use Case “Phản hồi upload tài liệu”

### Use Case download tài liệu

* + 1. **Use Case download tài liệu**

**Diagram

Description automatically generated**

*Hình 1.5: Lược đồ Use Case download tài liệu*

* + - 1. **Đặc tả Use Case download tài liệu**

|  |  |
| --- | --- |
| **Use Case ID** | 4 |
| **Tên Use Case** | Download tài liệu. |
| **Mô tả Use Case** | Use Case này cho phép người dùng download tài liệu về máy để sử dụng. |
| **Actor chính** | Người dùng (User). |
| **Actor phụ** | Không |
| **Tiền điều kiện** | User đã đăng nhập. |
| **Hậu điều kiện** | Không có. |

|  |  |
| --- | --- |
| **Luồng hoạt động** | 1. Người dùng chọn tài liệu cần download. 2. Nhấn vào button download và chờ tài liệu tải về máy. |
| **Luồng thay thế** | Không có. |
| **Luồng ngoại lệ** | Không có. |

*Bảng 1.4: Đặc tả Use Case “Download tài liệu”*

* + 1. **Use Case quản lý tài khoản**

**Diagram

Description automatically generated**

*Hình 1.6: Lược đồ Use Case quản lý tài khoản*

* + - 1. **Đặc tả Use Case quản lý tài khoản**

|  |  |
| --- | --- |
| **Use Case ID** | 5 |
| **Tên Use Case** | Quản lý tài khoản |
| **Mô tả Use Case** | Use Case này cho phép người dùng xem thông tin tài khoản, chỉnh sửa thông tin, mật khẩu. |
| **Actor chính** | Người dùng (User). |
| **Actor phụ** | Không |
| **Tiền điều kiện** | User đã đăng nhập. |
| **Hậu điều kiện** | Không có. |
| **Luồng hoạt động** | 1. Người dùng nhấn vào biểu tượng tài khoản. 2. Hệ thống sẽ hiển thị các lựa chọn để thao tác với tài khoản. 3. Người dùng chọn options cần thao tác và tiến hành sử dụng. |
| **Luồng thay thế** | Không có. |
| **Luồng ngoại lệ** | Không có. |

*Bảng 1.5: Đặc tả Use Case “Quản lý tài khoản”*

* + 1. **Use Case xem tài liệu**

**Diagram

Description automatically generated**

*Hình 1.7: Lược đồ Use Case xem tài liệu*

* + - 1. **Đặc tả Use Case xem tài liệu**

|  |  |
| --- | --- |
| **Use Case ID** | 6 |
| **Tên Use Case** | Xem tài liệu |

|  |  |
| --- | --- |
| **Mô tả Use Case** | Use Case này cho phép người dùng xem tài liệu dưới dạng slide trình chiếu trước khi download. Hoặc xem lại tài liệu sau khi upload để chỉnh sửa nếu cần. |
| **Actor chính** | Người dùng (User). |
| **Actor phụ** | Không |
| **Tiền điều kiện** | Không có. |
| **Hậu điều kiện** | Không có. |
| **Luồng hoạt động** | 1. Người dùng chọn vào tài liệu muốn download hoặc đã upload. 2. Hệ thống sẽ hiển thị một khung dưới dạng slide người dùng có thể nhấn previous or next để xem nội dung các slide tài liệu. |
| **Luồng thay thế** | Không có. |
| **Luồng ngoại lệ** | Không có. |

*Bảng 1.6: Đặc tả Use Case “Xem tài liệu”*

* 1. **Lược đồ tuần tự**
     1. **Lược đồ tuần tự chức năng đăng nhập**

**Diagram

Description automatically generated**

*Hình 1.7: Lược đồ tuần tự chức năng “Đăng nhập”*

* + 1. **Lược đồ tuần tự chức năng đăng kí**

**Diagram

Description automatically generated**

*Hình 1.8: Lược đồ tuần tự chức năng “Đăng kí”*

* + 1. **Lược đồ tuần tự chức năng upload tài liệu**

**Diagram

Description automatically generated**

*Hình 1.9: Lược đồ tuần tự chức năng “Upload tài liệu”*

* + 1. **Lược đồ tuần tự chức năng download tài liệu**

**Diagram

Description automatically generated**

*Hình 1.10: Lược đồ tuần tự chức năng “Download tài liệu”*

* + 1. **Lược đồ tuần tự chức năng quản lý tài khoản**

**Diagram

Description automatically generated**

*Hình 1.11: Lược đồ tuần tự chức năng “Quản lý tài khoản”*

* 1. **Thiết kế cơ sở dữ liệu**
     1. **Lược đồ cơ sở dữ liệu tổng quát**

Diagram, schematic

Description automatically generated

*Hình 1.12: Lược đồ cơ sở dữ liệu tổng quát*

* 1. **Các API của hệ thống**
     1. **API chức năng Comment**

Graphical user interface, application

Description automatically generated

* + 1. **API của Document**

Gồm các API xem, thêm, sửa, xóa, tài liệu upload hoặc tài liệu download.

Graphical user interface, application

Description automatically generated

* + 1. **API chức năng Like**

Background pattern

Description automatically generated with low confidence

* + 1. **API của Subject**

Gồm các API xem, thêm, xóa, sửa các chủ đề của tài liệu.

Background pattern

Description automatically generated

* + 1. **API của User**

Gồm các API của chức năng đăng nhập, đăng kí người dùng.

A picture containing application

Description automatically generated

* 1. **Giao diện của hệ thống**
     1. **Giao diện đăng nhập**

Graphical user interface, application

Description automatically generated

* + 1. **Giao diện đăng kí**

Graphical user interface

Description automatically generated

* + 1. **Giao diện Upload tài liệu**

Graphical user interface

Description automatically generated with medium confidence

* + 1. **Giao diện Download tài liệu**

Graphical user interface, text, application

Description automatically generatedText

Description automatically generated

* + 1. **Giao diện xem tài liệu**

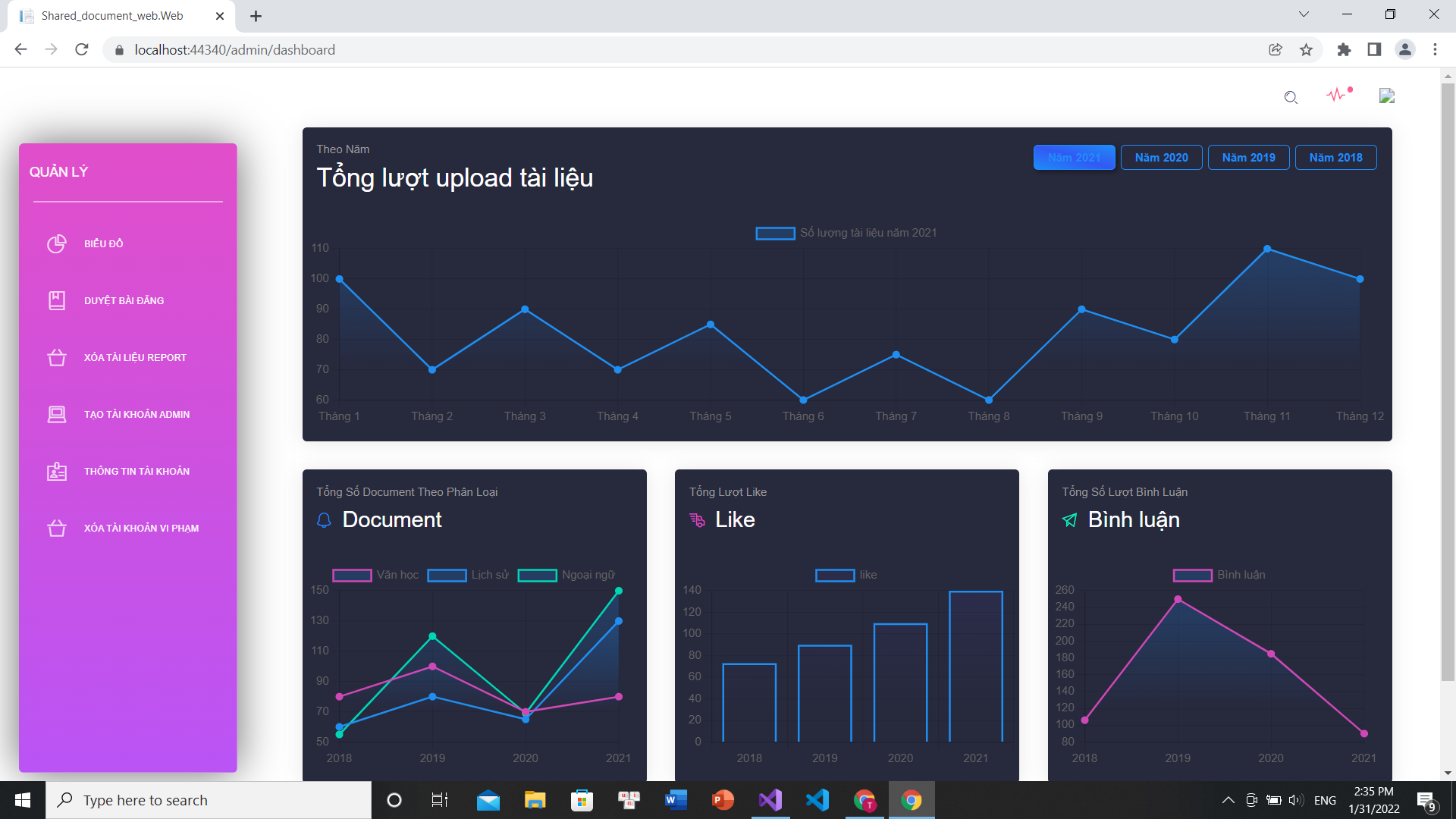
Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

* + 1. **Giao diện thống kê lượt truy cập của Admin**



# GIỚI THIỆU TỔNG QUAN VỀ REACTJS

Với đề tài được giao là tạo ra một trang web chia sẻ tài liệu, chúng em sử dụng .NET Core làm backend kết hợp với ReactJs phía forntend, tất cả đều được thực hiện trên công cụ Visual Studio kết hợp với ngôn ngữ Java.

Các API phía backend được hiện thị thông qua url “/swagger”. Bởi vì phần backend .NET Core được học trực tiếp từ môn học và bắt buộc phải có trong project, báo cáo dưới đây chúng em chỉ bao gồm các phần về ReactJS.

## Giới thiệu về React

React được Facebook định nghĩa chính thức vào năm như một thư viện JavaScript để xây dựng thư viện người dùng ( A JavaScript library for building user interfaces). React có thể xây dựng website sử dụng JavaScript hoàn toàn để thao tác với HTML thông qua kỹ thuật VirtualDOM, cho phép người lập trình có thể tạo ra các dự án web lớn, có giao diện ứng dụng phức tạp từ 1 mã nguồn nhỏ và độc lập được gọi là Component.

Không giống như Angular hay Vue, cũng không giống như một vài ý kiến trên các diễn đàn công nghệ rằng React là một framework, React là một thư viện, để thành một giải pháp hoàn chỉnh cho việc xây dựng ứng dụng, chúng ta cần kết hợp React với nhiều thư viện hơn. React chỉ tập trung vào xây dựng giao diện người dùng (building user interfaces), đồng nghĩa với chỉ tập trung vào rendering (kết xuất - nghĩa là xuất ra một thứ gì đó cho người dùng nhìn thấy) và đảm bảo nó được đồng bộ hóa với State của ứng dụng **(Reactjs thật đơn giản).**

Nhà lập trình có thể sử dụng React để mô tả giao diện người dùng và từ đó xây dựng nên giao diện người dùng thật sự - điều mà nếu xây dựng thủ công với các API web gốc bằng JavaScript sẽ khó khăn hơn nhiều lần. Các tuyên bố rằng “React là khai báo” (React is declarative) bắt đầu từ đó. React sẽ quản lý và dịch các mô tả khai báo mà developer đã lập trình thành giao diện người dùng thực tế trong trình duyệt. So với HTML (cũng là một loại ngôn ngữ trình báo), React phải khai báo cho các giao diện HTML, đại diện cho dữ liệu động, không chỉ nội dung tĩnh như HTML.

## Ưu điểm của React

### Khắc phục nhược điểm của framework

React không phải là một framework, điều này làm cho React có những ưu điểm riêng biệt. Framework phục vụ cho dự án lớn, đặc biệt cho các nhóm công ty khởi nghiệp. Khi làm việc với famework, nhiều thiết kế đã được làm sẵn, cho người dùng một con đường rõ ràng để tập trung việc xây dựng logic cho ứng dụng. Tuy nhiên, famework cũng có một vài nhược điểm gây khó khăn đối với các developer có kinh nghiệm làm việc trên các đoạn mã lớn, 2 nhược điểm tiêu biểu là:

* Không linh hoạt (mặc dù một số tuyên bố là vậy): developer phải viết các mã code theo một cách nhất định, khó thay đổi.
* Lớn và đầy đủ chức năng: điều này kiến các framework khó có thể điều chỉnh cho các trường hợp riêng biệt.

Các nhược điểm này được React khắc phục triệt để, React chỉ tập trung vào một thứ: cho phép developer khai báo các giao diện người dùng của họ và mô hình hình hóa trạng thái của các giao diện này.

### Ưu điểm khiến React trở nên phổ biến

* Cực kỳ hiệu quả: React tạo ra 1 dom ảo, nơi mà các component tồn tại thực sự trên đó. React chỉ tính toán những thay đổi nào được cập nhập lên DOM và thực hiện chúng, giảm chi phí cho những thao tác cần trên DOM. Ngoài ra, developer không cần phải xử lý API DOM.
* Dễ dàng hơn khi viết các đoạn code JS: Reactjs cho phép trộn lẫn giữa code HTML và JavaScript, các đoạn HTML có thể được thêm vào thông qua hàm render mà không cần thực hiện nối chuỗi.
* Dễ dàng test giao diện: vì virtual DOM được cài hoàn toàn bằng JavaScript, dẫn đến việc các test case có thể được thực hiện 1 cách dễ dàng.
* Hiệu năng cao với các dữ liệu thay đổi liên tục, dễ bảo trì, sửa lỗi.
* Tính tin cậy cao: nhóm React tại Facebook kiểm tra tất cả các cải tiến và tín năng mới được giới thiệu cho React ngay tại trên facebook.com, làm tăng sự tin cậy của thư viện này trong cộng đồng. Rất hiếm để tìm được các lỗi lớn và nghiêm trọng trong các bản phát hành của React vì nó chỉ được phát hành sau khi thông qua thử nghiệm

## Một số khái niệm cơ bản về React

### JSX

Đầu tiên, chúng ta có các biến như sau:

***const element = <h1>Word</h1>***

Cú pháp này không phải là 1 chuỗi kí tự, cũng không phải là một thẻ HTML. Đây là JSX, một cú phát mở rộng của JavaScript, được nhà phát triển khuyến khích sử dụng để mô tả giao diện. JSX đi kèm với toàn bộ tính năng của JavaScript, tạo ra các phần tử React.

Sử dụng JSX không phải điều bắt buộc trong React, nhưng phần lớn developer đều cho rằng JSX hữu dụng khi làm việc với giao diện trong mã JavaScript. Nó cũng cho phép React hiển thị những thông báo lỗi và “lời cảnh báo” hữu ích hơn.

### Component

Về cơ bản, Component là những hàm (function) hay lớp (class) có output là các thẻ ML, chia UI thành các phần độc lập, hoàn toàn tách biệt nhau, có thể tái sử dụng.

Về mặt khái niệm, Component nhận vào bất kì đầu vào nào (còn được gọi là “props”) sau đó trả về các React elements giống như các hàm JavaScript khác.

Component có thể được định nghĩa qua các hàm JavaScript hoặc thông qua việc sử dụng ES6 class.

### Props

Props (Properties), có thể được hiểu như những thuộc tính của 1 Component. Nếu coi Component như là đối tượng trong lập trình OOP thì Props giống như là những miêu tả về tính chất, đặc trưng của 1 đối tượng.

Điều quan trọng khi sử dụng props là, khi một component được định nghĩa dưới dạng function hoặc class, thì nó không được phép thay đổi props của chính nó. React có tính khả chuyển cao, nhưng nó vẫn có quy tắc riêng: mọi React components đều phải giống như các pure functions đối với props của chúng (các giá trị đầu vào không thay đổi và luôn trả về chỉ một kết quả).

### State

Trong React, State được dùng để biểu diễn trạng thái của Component. Tức là giá trị của State có thể thay đổi được, trái ngược hoàn toàn với Props là 1 giá trị bất biến. Trong khi props có thể được truyền sang các components khác nhau thì state chỉ tồn tại trong phạm vi của components chứa nó, mỗi khi state thay đổi thì components đó sẽ được render lại.

State được dùng để phản hồi các yêu cầu từ client, hoặc để lưu trữ một dữ liệu trong Component

Truy cập thuộc tính của đối tượng state: this.state.propertyName

### Vòng đời

Mỗi component dạng class của React có 3 giai đoạn chính

* Mounting: khi đặt một component vào DOM.
* Updating: khi component được update ( update khi có thay đổi trong props hoặc state).
* Unmounting: khi một component được gỡ khỏi cây DOM.

#### Mounting

Bao gồm các phương thức:

* **constructor()**: được gọi khi component được tạo ra. Có thể được dùng để tạo state cũng như “blind” các hàm cho component. Chú ý khi sử dụng: phải khi báo tham số props và gọi super(props) đầu tiên.
* **componentWillMount()**: gọi sau khi component được render. Được dùng để đăng ký các sự kiện toàn cục, tính toán và set lại state dựa vào các props
* **render()**:là phương thức bắt buộc và trả ra HTML cho DOM. Chú ý khi sử dụng: gây ra lặp vô hạn khi gọi setState() trong hàm này (vì mỗi lần set lại state hàm render sẽ được gọi).
* **static getDerivedStateFromProps():** gọi ngay trước khi render các thành phần trong DOM, được chạy khi mà props của component đã được sinh ra có sự thay đổi, là phiên bản update của hàm componentWillReceiveProps() với các cải tiến như có thể truy cập cả với nextProps và prevState trước đó mà với mà với phiên bản cũ chỉ có thể so sánh với nextProps.

#### Updating

Bao gồm các phương thức:

* **getDerivedStateFromProps():** gọi khi component được cập nhật.
* **shouldComponentUpdate():** trả về giá trị boolean cho biết React có tiếp tục render hay không, có mặc định là True, được gọi trước khi render.
* **render()** gọi khi component cập nhật render lại HTML cho DOM.
* **getSnapshotBeforeUpdate()** được gọi trước khi props và state được cập nhật.
* **componentDidUpdate()** gọi sau khi được cập nhật (ngay khi render lần thứ 2 trở đi), có thể được thông qua để thao các trên các phần tử DOM bằng JS.

#### Unmounting

* **componentWillUnmount():** gọi sau khi gỡ bỏ 1 component ra khỏi DOM, dùng để thực hiện các thao tác dọn dẹp (loại bỏ phần tử thừa, dọn các timer).

## 1.3.6 React Router

React Router là một thư viện dùng để định tuyến tiêu chuẩn trong React. Nhiệm vụ của React Router là giữ cho URL trình duyệt đồng bộ với giao diện của ứng dụng, cho phép định tuyến "luồng dữ liệu" của ứng dụng theo cách rõ ràng, giúp bạn khẳng định với đường dẫn này, bạn có router này và thể hiện UI tương ứng.

React Router được dùng chủ yếu để phát triển các ứng dụng Web dạng Single Page. Thông qua nó, các trang web có thể được di chuyển qua lại mà không cần phải nạp lại trang . React có 3 gói dành cho routing:

* react-router: bao gồm các thành phần chung giữa react-router-dom và react-router-native, cung cấp các thành phần routing chính cho ứng dụng.
* react-router-native: dùng cho các ứng dụng di động và React Native.
* react-router-dom: dùng cho các ứng dụng Web, có một số cải tiến cho trình duyệt so với React Router, đặc biệt với NavLink.

## 1.3.7 Hooks Effect

Hooks effect được dùng trong function, nơi mà vòng đời không thể hoạt động.

Hooks effect được sử dụng tương đương với các phương thức componentDidMount(), componentDidUpdate(), componentWillUnmount().

Phương thức áp dụng: useEffect().

## 1.3.8 React Axious

Axios là một HTTP client làm việc dựa trên promise được dùng để hỗ trợ cho việc cung cấp các API từ đơn giản đến phức tạp, dễ dàng sử dụng cho phía trình duyệt và NodeJS. Trong phần đồ án, Axious được sử dụng để kết nối Backend và Frontend.

Một vài option quan trọng trong React Axious:

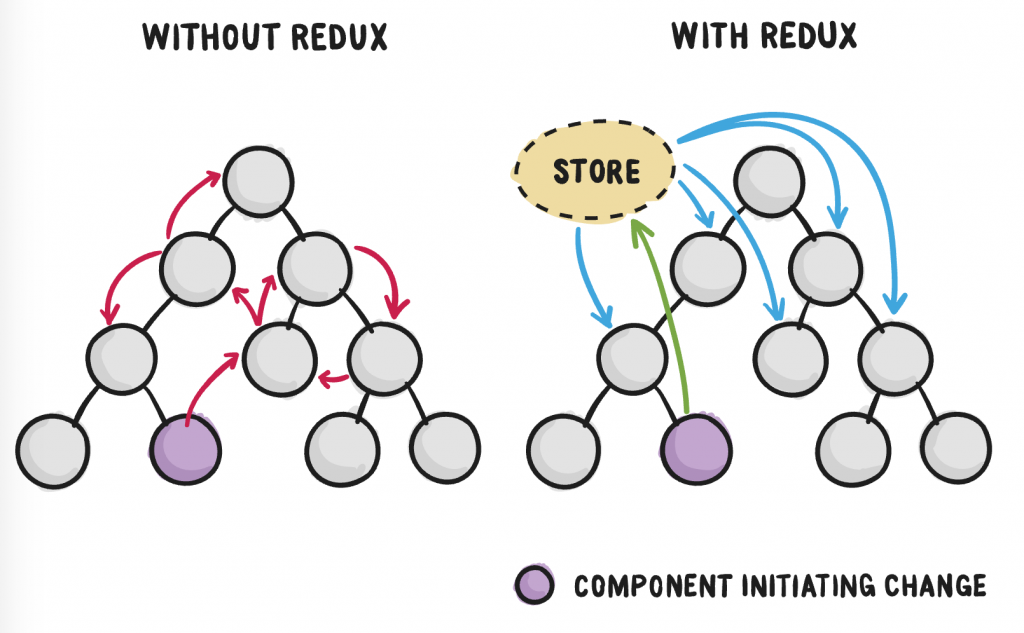
* instance: thể hiện của axios.
* url (bắt buộc): endpoint API.
* method (bắt buộc): phương thức HTTP request. (Vd: GET/POST/DELETE)
* data: các dữ liệu thể hiện trong request body.
* params: dữ liệu được serialize dính vào url dưới dạng query string.
* config: thông tin cấu hình cho axios.
* debounce: thời gian tối thiểu giữa các request.
* onSuccess: sự kiện được gọi khi request thực hiện thành công.
* onError: sự kiện được gọi khi request thực hiện có lỗi.
* onLoading: thực hiện khi bắt đầu request.

## 1.3.9 React Redux

Redux là một thư viện để quản lý trạng thái của ứng dụng và cập nhật state sử dụng các sự kiện gọi là actions.

Nó cung cấp một store trung tâm lưu trữ state thay vì lưu trữ trong 1 root component, store này được sử dụng trong toàn bộ ứng dụng.

Không thể thay đổi trực tiếp state của ứng dụng trong Redux, để thay đổi state trong redux phải phát một action, đối tượng mô tả những gì xảy ra.



Hình 1.1: Quá trình change data của event khi có và không có Redux

#### Action

Một action có thể được mô tả dưới dạng một plain objec JS, tất cả action đều có thuộc tính type , đóng vai trò mô tả action.

Action cũng có thể chứa thêm các thông tin khác ngoài trường bắt buộc là trường Type, quy ước những thông tin này đặt trong thuộc tính payload của đối tượng action hoàn toàn dựa vào sự quyết định của người lập trình, những đối tượng bổ sung này còn được gọi là payload.

#### Chủ đề cấp độ 4

#### Reducer

Reducer nhận vào 2 tham số: state và action. Reducer là nơi duy nhất được thực hiện cập nhật state trong Redux, được dùng để nhận state hiện tại và đối tượng action để cập nhật state khi cần thiết và trả về state mới thông qua Action.

## 1.3.8.3 Store

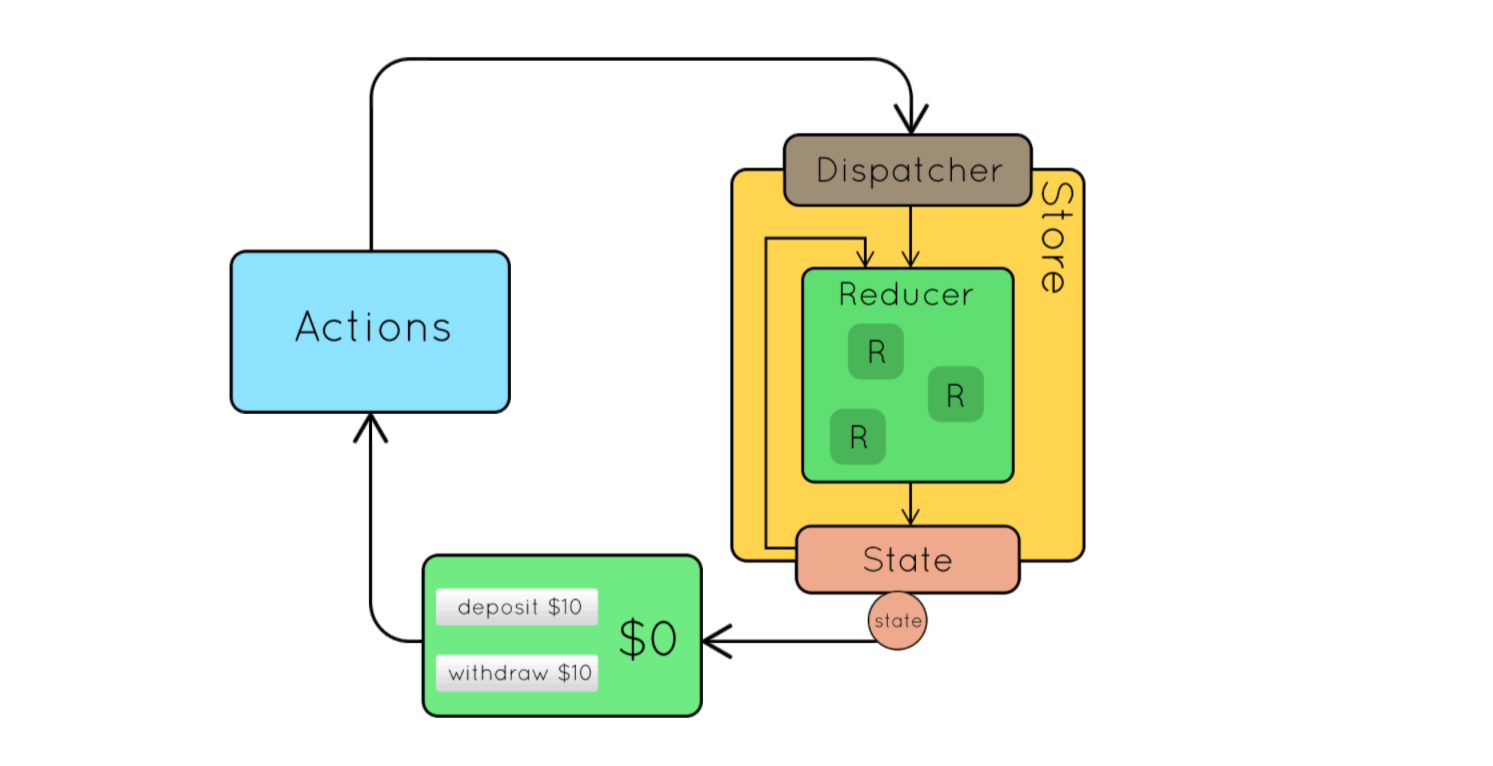
Nơi duy nhất chứa state hiện hành của ứng dụng. Phương thức tạo store.

**createStore(reducer, [preloadedState], [enhancer])**

Một vài phương thức trong Store:

* getState: trả về giá trị state hiện tại.
* subscribe: đăng ký hàm callback được gọi mỗi khi store được cập nhật.
* dispatch: được gọi, store chạy reducer nhận state được cập nhật và chạy các hàm callback được subscribe để cập nhật UI.

## 1.3.8.4 Cơ chế hoạt động



Hình 1.1: Mô hình hoạt động của Redux

Một action xảy ra khi user tương tác với ứng dụng (nhấn vào nút deposit $10). Hàm reducer được gọi với state hiện tại truyền đến action, action được truyền vào và trả về state mới.

Store thông báo các view thực thi bằng cách gọi các hàm callback của chúng.

React component đọc dữ liệu từ store và truyền action tới store để cập nhật dữ liệu.

Redux theo dõi (subscribe) store để kiểm tra dữ liệu của component nào thay đổi sẽ nạp lại (rerender) component đó. (Sau khi Redux xử lý, trả về 1 State và khởi tạo lại giá trị của $0 thành $10).