

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC VĂN HIẾN
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO HỌC PHẦN
TTTN CÔNG NGHỆ THÔNG TIN**

Đề tài:

**XÂY DỰNG WEBSITE
MẠNG XÃ HỘI SHAREFUN**

GVHD: ThS. NGUYỄN MINH THI

SVTH: HÀ VĂN ĐƯỢC

MSSV: 181A010208

LỚP: INT55101

TP. HỒ CHÍ MINH – 2022

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC VĂN HIẾN
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO HỌC PHẦN
TTTN CÔNG NGHỆ THÔNG TIN**

Đề tài:

**XÂY DỰNG WEBSITE
MẠNG XÃ HỘI SHAREFUN**

GVHD: ThS. NGUYỄN MINH THI

SVTH: HÀ VĂN ĐƯỢC

MSSV: 181A010208

LỚP: INT55101

TP. HỒ CHÍ MINH – 2022

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Họ và tên giảng viên: **NGUYỄN MINH THI**

Tên đề tài: **XÂY DỰNG WEBSITE MẠNG XÃ HỘI SHAREFUN.**

Nội dung nhận xét:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:

Bảng số:

Bảng chữ:

GIẢNG VIÊN CHẤM

(Ký, ghi rõ họ tên)

NGUYỄN MINH THI

LỜI CẢM ƠN

Trước hết em xin chân thành cảm ơn Thầy Nguyễn Minh Thi - giảng viên Bộ môn Công Nghệ Thông Tin Trường Đại học Văn Hiến, đã tận tình hướng dẫn giúp đỡ em trong suốt quá trình nghiên cứu và hoàn thành công việc thực tập tốt nghiệp cũng như xây dựng báo cáo thực tập tốt nghiệp.

Cảm ơn anh Nguyễn Thành Quang – Phó giám đốc Công ty TNHH Quốc tế Xanh tươi sáng đã trực tiếp giúp đỡ em trong công việc tìm hiểu đề tài.

Cuối cùng em xin chân thành cảm ơn các Thầy cô khoa Công nghệ thông tin Trường Đại học Văn Hiến đã trực tiếp giảng dạy em trong những năm học vừa qua, cùng bạn bè người thân đã nhiệt tình ủng hộ, động viên em trong suốt quá trình học tập và xây dựng đề tài thực tập tốt nghiệp.

Em xin chân thành cảm ơn!

TP.Hồ Chí Minh, ngày 18 tháng 12 năm 2022

SINH VIÊN THỰC HIỆN

HÀ VĂN ĐƯỢC

DANH MỤC HÌNH

Hình 1: Cơ chế One-way data binding	4
Hình 2: Sơ đồ cấu trúc dự án Expressjs.....	5
Hình 3: Quy trình làm việc của mô hình MVC	9
Hình 4: Mô hình mạng máy tính Client-Server.....	11
Hình 5: Cấu trúc hệ thống ShareFun	11
Hình 6: Cấu trúc thành phần pages.....	12
Hình 7: Cấu trúc thành phần Layouts.....	12
Hình 8: Cấu trúc thành phần components	13
Hình 9: Cấu trúc thành phần Redux	14
Hình 10: Cấu trúc server.....	14
Hình 11: Cấu trúc thành phần middlewares	15
Hình 12: Code preview authRoute	16
Hình 13: Code preview postRoute	16
Hình 14: Code preview userRoute	17
Hình 15: Quy trình làm việc của Back-end.....	17
Hình 16: Code preview Get Current User trong authController	18
Hình 17: Code preview LoginController trong authController.....	18
Hình 18: Code preview RegisterController trong authController.....	19
Hình 19: Code preview getAllPost controller	20
Hình 20: Code preview getAPost controller	20
Hình 21: Code preview createAPost controller.....	21
Hình 22: Code preview updateAPost controller.....	21
Hình 23: Code preview deleteAPost controller.....	22
Hình 24: Code preview Follow A User controller	22
Hình 25: Quy trình đăng ký.....	23
Hình 26: Quy trình đăng nhập	24
Hình 27: Quy trình Tạo bài post.....	25
Hình 28: Giao diện Modal Login	26
Hình 29: Giao diện Modal đăng ký	26
Hình 30: Lỗi đăng ký "Tên đăng nhập đã tồn tại"	27
Hình 31: Lỗi đăng ký "Mật khẩu không trùng nhau"	27

Hình 32: Lỗi đăng nhập "Sai mật khẩu"	28
Hình 33: Lỗi đăng nhập "Tài khoản/ email chưa được đăng ký"	28
Hình 34: Giao diện tính năng đăng xuất.....	29
Hình 35: Giao diện Trang chủ	30
Hình 36: Giao diện Trang profile	31
Hình 37: Giao diện Trang tạo bài post	32

DANH MỤC THUẬT NGỮ - KÝ TỰ

STT	Thuật ngữ	Nghĩa từ
1	User	Người dùng
2	Post	Bài đăng
3	Structure	Cấu trúc
4	Component	Thành phần
5	Model	Tầng dữ liệu
6	Contronller	Bộ điều khiển
7	View	Tầng giao diện
8	API	Phương thức trung gian kết nối các ứng dụng và thư viện khác nhau
9	Render	Cập nhật hiển thị lại UI
10	Library	Thư viện source code
11	Route	Đường dẫn, tuyến đường
12	JWT	JSON Web Token
13	Modal	Hộp thoại
14	Popup	Cửa sổ hiện lên

MỤC LỤC

Chương 1: MỞ ĐẦU	1
1.1. Lý do chọn đề tài	1
1.2. Mục tiêu nghiên cứu	1
1.3. Nội dung thực tập	1
1.3.1. Quá trình thực tập	1
1.3.2. Địa điểm thực tập	2
1.4. Cấu trúc bài báo cáo	2
Chương 2: TỔNG QUAN VỀ CÔNG NGHỆ ÁP DỤNG	3
2.1. MERN stack	3
2.1.1. React	3
2.1.2. ExpressJS	5
2.1.3. NodeJS	6
2.1.4. MongoDB	6
2.2. Mô hình MVC	8
Chương 3: PHÂN TÍCH – THIẾT KẾ HỆ THỐNG	10
3.1. Mô tả hệ thống	10
3.2. Mô hình mạng máy tính	10
3.3. Cấu trúc chương trình (Program structure)	11
3.4. Thiết kế cấu trúc dữ liệu (Model)	15
3.5. Thiết kế API chuẩn RESTful (Route)	16
3.6. Thiết kế Controller tương ứng với từng API	17
3.7. Quy trình đăng ký, đăng nhập	23
3.8. Quy trình đăng nhập	24
3.8. Quy trình Tạo/Chỉnh sửa/Xóa/Lấy post	25
Chương 4: THIẾT KẾ GIAO DIỆN WEBSITE	26
4.1. Modal đăng nhập, đăng ký	26
4.2. Trang timeline (trang chủ)	30
4.3. Trang profile (hồ sơ cá nhân)	31
4.4. Trang tạo bài post	32
Chương 5: TỔNG KẾT THÀNH TỰU VÀ HƯỚNG PHÁT TRIỂN	33
5.1. Thành tựu đạt được	33
5.2. Hạn chế	33
5.3. Hướng phát triển	33

Chương 1: MỞ ĐẦU

1.1. Lý do chọn đề tài

Ngày nay, chúng ta đang được sống trong kỷ nguyên của tin học nhờ sự vượt bậc, sự bùng nổ mạnh mẽ của công nghệ thông tin. Công nghệ thông tin không chỉ dừng lại ở mục đích phục vụ cho khoa học kỹ thuật mà đi sâu vào đời sống, chính trị, kinh tế, xã hội trở nên thân thiện, gần gũi, mang lại nhiều lợi ích cho con người. Công nghệ thông tin ngày càng khẳng định được tính hữu dụng và sức mạnh trong mọi phương diện, mọi ngành nghề của cuộc sống, nhất là trong thời đại kinh tế thị trường hiện đại như bây giờ.

Đi kèm theo đó, các hệ thống website cũng phát triển không ngừng với đủ các lĩnh vực: văn hóa, thời sự, khoa học công nghệ, làm đẹp, nấu ăn, thể thao, ca nhạc, phim ảnh, giáo dục, y tế... Tuy nhiên, để tạo ra một trang web có sức sống bền lâu thì bản thân nó phải mang lại lợi ích cho nhiều người. Giao diện bắt mắt là yếu tố quan trọng để người xem đến website của bạn. Nhưng yếu tố quyết định giữ chân độc giả lại là nội dung mà website cung cấp.

Sau khi tìm hiểu và xin ý kiến của giảng viên hướng dẫn, em đã lựa chọn đề tài “XÂY DỰNG WEBSITE MẠNG XÃ HỘI SHAREFUN”

1.2. Mục tiêu nghiên cứu

Thiết kế, xây dựng một website mạng xã hội nơi mọi người có thể giao lưu, chia sẻ những niềm vui với bạn bè, người thân, gia đình và xã hội.

1.3. Nội dung thực tập

1.3.1. Quá trình thực tập

- Tìm hiểu cách quản lý và quy trình làm việc của công ty
- Tiếp cận thực tế và học hỏi cách làm việc của công ty
- Thực hiện những công việc mà công ty giao phó
- Thu thập tài liệu để viết báo cáo thực tập

1.3.2. Địa điểm thực tập

CÔNG TY TNHH QUỐC TẾ XANH TƯƠI SÁNG có ngành nghề kinh doanh chính là “Buôn bán máy móc, thiết bị và phụ tùng máy khác”. Có trụ sở tại 967B Nguyễn Xiển, Phường Long Bình, Quận 9, TP Hồ Chí Minh

1.4. Cấu trúc bài báo cáo

Bài báo cáo gồm 5 chương:

Chương 1: Mở đầu

Chương 2: Tổng quan về các công nghệ áp dụng

Chương 3: Phân tích – thiết kế hệ thống

Chương 4: Thiết kế giao diện website

Chương 5: Tổng kết thành tựu và hướng phát triển

Chương 2: TỔNG QUAN VỀ CÔNG NGHỆ ÁP DỤNG

2.1. MERN stack

MERN là một thuật ngữ rút gọn chỉ tổ hợp open source các công nghệ đều liên quan đến javascript hot nhất hiện nay: MongoDB, ExpressJS, React/React Native, NodeJS. MERN stack là một stack Javascript được thiết kế để giúp phát triển ứng dụng web toàn diện xấp xỉ dễ dàng hơn và nhanh hơn.

Trong đó, tất cả bốn công nghệ này cung cấp một khuôn khổ hoàn chỉnh cho các nhà phát triển để tạo ra bất kỳ ứng dụng web nào. MERN đang tuân theo kiến trúc 3 tầng truyền thống, bao gồm tầng hiển thị front-end (React.js), tầng ứng dụng (Express.js và Node.js) và tầng cơ sở dữ liệu (MongoDB).

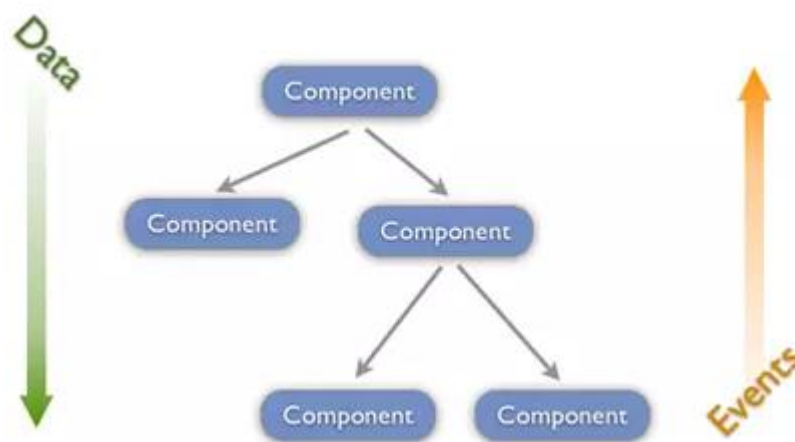
2.1.1. React

React hay còn có tên gọi ReactJS là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code javascript trong code html thông qua các attribute như ng-model, ng-repeat...thì với react là một library cho phép nhúng code html trong code javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong JS. Tích hợp giữa javascript và HTML vào trong JSX làm cho các component dễ hiểu hơn.

React là một thư viện UI phát triển tại Facebook để hỗ trợ việc xây dựng những thành phần (components) UI có tính tương tác cao, có trạng thái và có thể sử dụng lại được. React được sử dụng tại Facebook trong production, và www.instagram.com được viết hoàn toàn trên React.

Một trong những điểm hấp dẫn của React là thư viện này không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau. React so sánh sự thay đổi giữa các giá trị của lần render này với lần render trước và cập nhật ít thay đổi nhất trên DOM.

React sử dụng cơ chế one-way data binding – luồng dữ liệu 1 chiều. Dữ liệu được truyền từ parent đến child thông qua props. Luồng dữ liệu đơn giản giúp ta dễ dàng kiểm soát cũng như sửa lỗi.



Hình 1: Cơ chế One-way data binding

Với đặc điểm ở trên, React dùng để xây dựng các ứng dụng lớn mà dữ liệu của chúng thay đổi liên tục theo thời gian. Dữ liệu thay đổi thì hầu hết kèm theo sự thay đổi về giao diện. Ví dụ như Facebook: trên Newsfeed của bạn cùng lúc sẽ có các status khác nhau và mỗi status lại có số like, share, comment liên tục thay đổi. Khi đó React sẽ rất hữu ích để sử dụng.

Giới thiệu về JSX

JSX là một dạng ngôn ngữ cho phép viết các mã HTML trong Javascript. Đặc điểm: Faster: Nhanh hơn. JSX thực hiện tối ưu hóa trong khi biên dịch sang mã Javascript. Các mã này cho thời gian thực hiện nhanh hơn nhiều so với một mã tương đương viết trực tiếp bằng Javascript. Safer: an toàn hơn. Ngược với Javascript, JSX là kiểu statically-typed, nghĩa là nó được biên dịch trước khi chạy, giống như Java, C++. Vì thế các lỗi sẽ được phát hiện ngay trong quá trình biên dịch. Ngoài ra, nó cũng cung cấp tính năng gỡ lỗi khi biên dịch rất tốt. Easier: Dễ dàng hơn. JSX kế thừa dựa trên Javascript, vì vậy rất dễ dàng để cho các lập trình viên Javascripts có thể sử dụng.

Giới thiệu về Components

React được xây dựng xung quanh các component, chứ không dùng template như các framework khác. Trong React, chúng ta xây dựng trang web sử dụng những thành phần

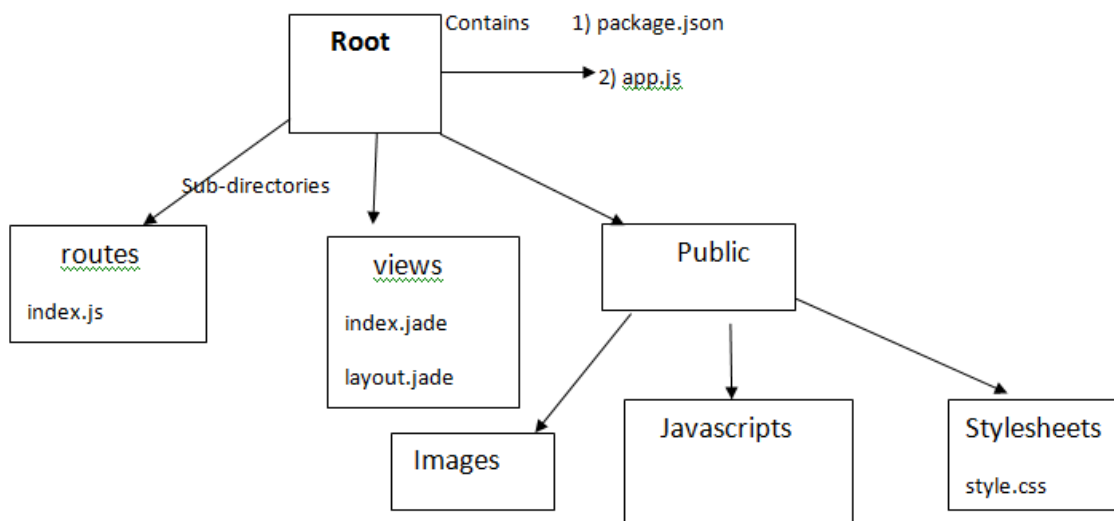
(component) nhỏ. Chúng ta có thể tái sử dụng một component ở nhiều nơi, với các trạng thái hoặc các thuộc tính khác nhau, trong một component lại có thể chứa thành phần khác. Mỗi component trong React có một trạng thái riêng, có thể thay đổi, và React sẽ thực hiện cập nhật component dựa trên những thay đổi của trạng thái. Mọi thứ React đều là component. Chúng giúp bảo trì mã code khi làm việc với các dự án lớn. Một react component đơn giản chỉ cần một method render. Có rất nhiều methods khả dụng khác, nhưng render là method chủ đạo.

Props và State

Props: giúp các component tương tác với nhau, component nhận input gọi là props, và trả thuộc tính mô tả những gì component con sẽ render. Prop là bất biến. State: thể hiện trạng thái của ứng dụng, khi state thay đổi thì component đồng thời render lại để cập nhật UI.

2.1.2. ExpressJS

Expressjs là một framework được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. Expressjs hỗ trợ các method HTTP và middleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.



Hình 2: Sơ đồ cấu trúc dự án Expressjs

Tổng hợp một số chức năng chính của Expressjs như sau:

- Thiết lập các lớp trung gian để trả về các HTTP request.

- Define router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL.
- Cho phép trả về các trang HTML dựa vào các tham số.

2.1.3. NodeJS

NodeJS là một môi trường runtime chạy JavaScript đa nền tảng và có mã nguồn mở, được sử dụng để chạy các ứng dụng web bên ngoài trình duyệt của client. Nền tảng này được phát triển bởi Ryan Dahl vào năm 2009, được xem là một giải pháp hoàn hảo cho các ứng dụng sử dụng nhiều dữ liệu nhờ vào mô hình hướng sự kiện (event-driven) không đồng bộ.

Ưu điểm

- IO hướng sự kiện không đồng bộ, cho phép xử lý nhiều yêu cầu đồng thời.
- Sử dụng JavaScript – một ngôn ngữ lập trình dễ học.
- Chia sẻ cùng code ở cả phía client và server.
- NPM(Node Package Manager) và module Node đang ngày càng phát triển mạnh mẽ.
- Cộng đồng hỗ trợ tích cực.
- Cho phép stream các file có kích thước lớn.

Nhược điểm

- Không có khả năng mở rộng, vì vậy không thể tận dụng lợi thế mô hình đa lõi trong các phần cứng cấp server hiện nay.
- Khó thao tác với cơ sở dữ liệu quan hệ.
- Mỗi callback sẽ đi kèm với rất nhiều callback lồng nhau khác.
- Cần có kiến thức tốt về JavaScript.
- Không phù hợp với các tác vụ đòi hỏi nhiều CPU.

2.1.4. MongoDB

MongoDB Là một noSQL database hot nhất hiện nay. MongoDB thường đi với Mongoose – một library để giao tiếp với MongoDB dễ dàng hơn.

MongoDB là gì?

- MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng.
- MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.
- Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là collection thay vì bảng
- So với RDBMS thì trong MongoDB collection ứng với table, còn document sẽ ứng với row , MongoDB sẽ dùng các document thay cho row trong RDBMS.
- Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.
- Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB

Ưu điểm của MongoDB

- Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu, nên bạn muốn gì thì cứ insert vào thoải mái.
- Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau, không có join như trong RDBMS nên khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.
- MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node vào vào cluster:
- Trường dữ liệu “_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
- Khi có một truy vấn dữ liệu, bản ghi được cached lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.

Nhược điểm của mongoDB

- Một ưu điểm của MongoDB cũng chính là nhược điểm của nó. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.
- Tổn bộ nhớ do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên dễ bị dư thừa dữ liệu.
- Khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng điều này sẽ là nhược điểm vì sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện...

2.2. Mô hình MVC

MVC là viết tắt của cụm từ “Model-View-Controller“. Đây là mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau và mỗi thành phần đều có một nhiệm vụ riêng của nó và độc lập với các thành phần khác. Tên gọi 3 thành phần:

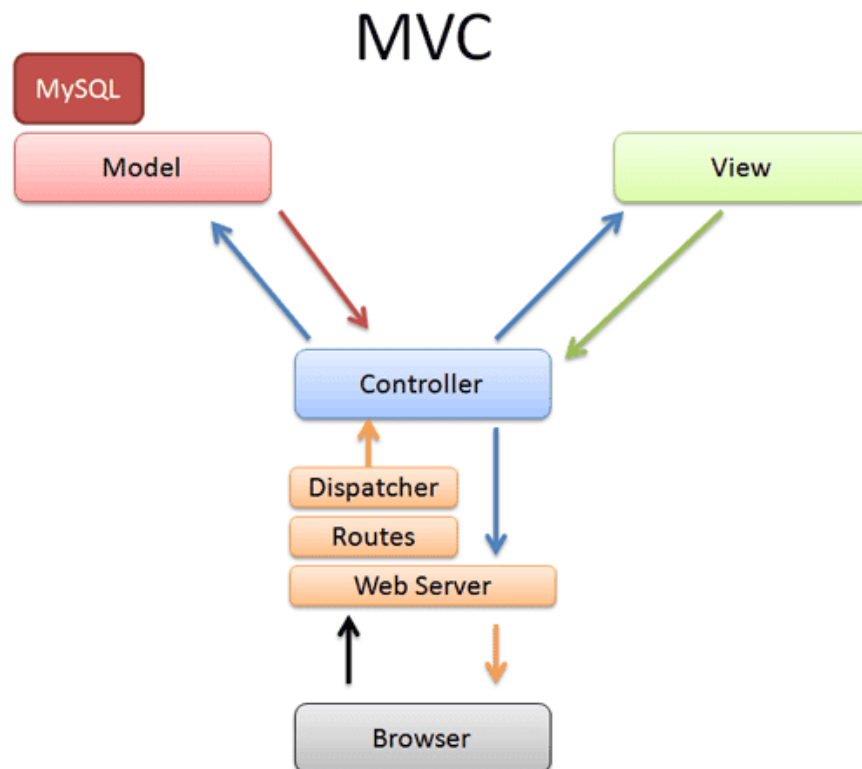
- Model (dữ liệu): Quản lý xử lý các dữ liệu.
- View (giao diện): Nói hiển thị dữ liệu cho người dùng.
- Controller (bộ điều khiển): Điều khiển sự tương tác của hai thành phần Model và View.

Mô hình MVC (MVC pattern) thường được dùng để phát triển giao diện người dùng. Nó cung cấp các thành phần cơ bản để thiết kế một chương trình cho máy tính hoặc điện thoại di động, cũng như là các ứng dụng web.

Model: Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng và là cầu nối giữa 2 thành phần bên dưới là View và Controller. Một model là dữ liệu được sử dụng bởi chương trình. Đây có thể là cơ sở dữ liệu, hoặc file XML bình thường hay một đối tượng đơn giản. Chẳng hạn như biểu tượng hay là một nhân vật trong game.

View: Đây là phần giao diện (theme) dành cho người sử dụng. View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.

Controller: Là bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View. Một controller bao gồm cả Model lẫn View. Nó nhận input và thực hiện các update tương ứng.



Hình 3: Quy trình làm việc của mô hình MVC

Luồng xử lý trong MVC

- Khi một yêu cầu của từ máy khách (Client) gửi đến Server. Thì bị Controller trong MVC chặn lại để xem đó là URL request hay sự kiện.
- Sau đó, Controller xử lý input của user rồi giao tiếp với Model trong MVC.
- Model chuẩn bị data và gửi lại cho Controller.
- Cuối cùng, khi xử lý xong yêu cầu thì Controller gửi dữ liệu trở lại View và hiển thị cho người dùng trên trình duyệt.

Ở đây, View không giao tiếp trực tiếp với Model. Sự tương tác giữa View và Model sẽ chỉ được xử lý bởi Controller.

Chương 3: PHÂN TÍCH – THIẾT KẾ HỆ THỐNG

3.1. Mô tả hệ thống

Mạng xã hội là một nền tảng trực tuyến, là nơi mà mọi người có thể xây dựng các mối quan hệ ảo với những người có chung tính cách, sở thích, nghề nghiệp... hoặc với cả những người có mối quan hệ ngoài đời thực.

Mạng xã hội hiện nay có nhiều dạng thức và tính năng khác nhau, và có thể truy cập dễ dàng từ nhiều phương tiện, thiết bị như máy tính bảng, laptop, điện thoại di động,...

Mạng xã hội tuy tồn tại dưới nhiều hình mô hình khác nhau nhưng nhìn chung, mạng xã hội đều có những điểm chung sau:

- Mạng xã hội là một ứng dụng được sử dụng trên nền tảng Internet.
- Tất cả nội dung trên mạng xã hội đều do người dùng tự tạo ra, tự chia sẻ.
- Mỗi người dùng trên mạng xã hội đều phải tạo tài khoản, hồ sơ riêng.
- Mạng xã hội sẽ kết nối tài khoản người dùng đến các tài khoản cá nhân, tổ chức khác thông qua các tài khoản ảo do người dùng tạo ra.

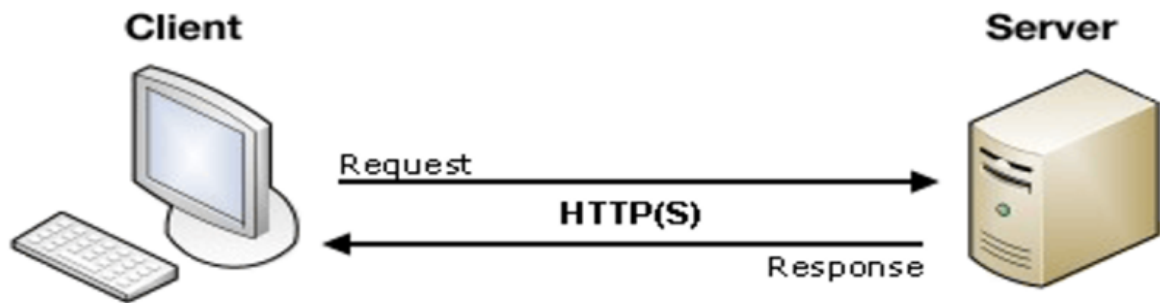
Yêu cầu chức năng:

- Người dùng có thể tạo tài khoản và thiết lập hồ sơ cá nhân
- Các tính năng đăng nhập, đăng xuất website
- Tính năng tạo bài đăng (post) cũng như có thể xem (get), chỉnh sửa (update), xóa (delete) theo yêu cầu
- Ngoài ra, các tính năng tương tác như like, bình luận, chia sẻ, theo dõi người dùng...

3.2. Mô hình mạng máy tính

Hệ thống của website Mạng xã hội ShareFun được xây dựng dựa trên mô hình mạng Client-Server gồm có 2 thành phần chính đó là máy khách (client) và máy chủ (server). Server chính là nơi lưu trữ tài nguyên cũng như cài đặt các chương trình dịch vụ theo đúng

yêu cầu của client. Ngược lại, Client bao gồm máy tính cũng như các thiết bị điện tử nói chung sẽ tiến hành yêu cầu đến server.

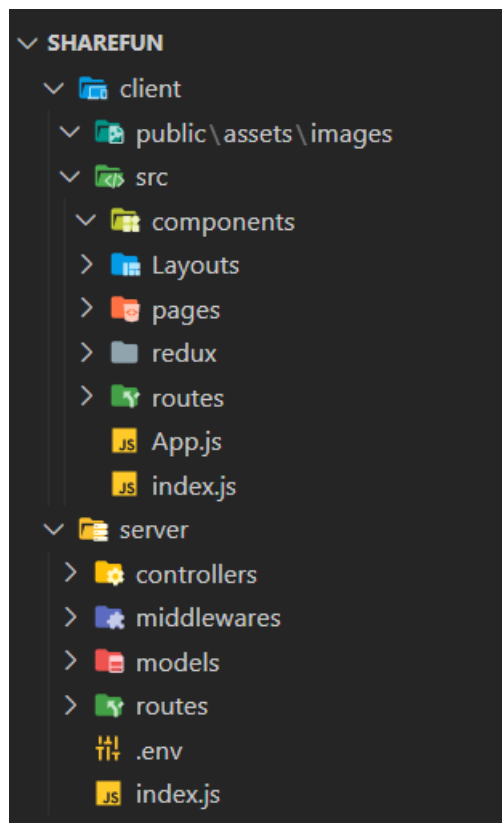


Hình 4: Mô hình mạng máy tính Client-Server

Nguyên lý hoạt động của mô hình

Client chính là khách hàng sử dụng dịch vụ muốn biết một thông tin cụ thể, là người đưa ra yêu cầu (request). Request đó sẽ được gửi đến server Server hay là nhà cung cấp thông tin, người tiếp nhận yêu cầu sẽ xử lý nó và cuối cùng sẽ gửi phản hồi (response) về client và client sẽ xử lý kết quả sau đó hiển thị ra màn hình.

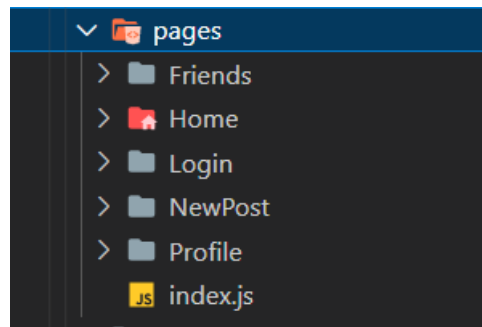
3.3. Cấu trúc chương trình (Program structure)



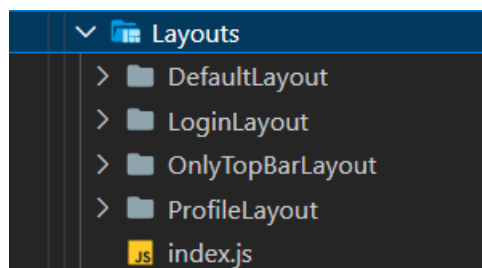
Hình 5: Cấu trúc hệ thống ShareFun

Trước hết, ta sẽ phân tích về cấu trúc của client

- + **client**: tập trung xử lý hiển thị thông tin, gửi các request đến server
- + **public**: thư mục chứa toàn bộ dữ liệu được phép công khai
- + **src**: thư mục gốc
- + **pages**: chứa các trang của hệ thống như trang đăng nhập, trang chủ, trang hồ sơ cá nhân, trang bạn bè...

*Hình 6: Cấu trúc thành phần pages*

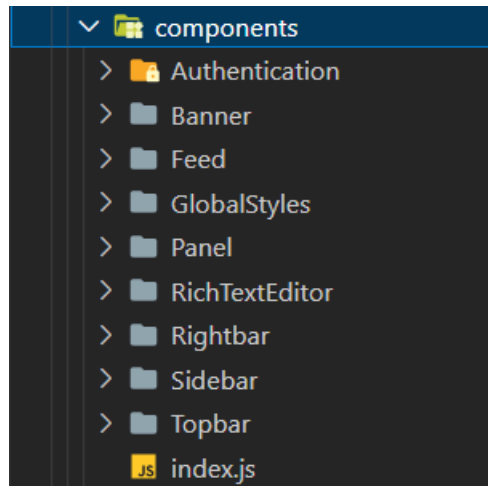
- + **layouts**: bố cục của website

*Hình 7: Cấu trúc thành phần Layouts*

- DefaultLayout là layout chính cho toàn bộ website gồm các thành phần: Header, Sidebar, Rightbar, Feed.
- LoginLayout không như DefaultLayout không bao gồm các thành phần nào
- ProfileLayout bao gồm một Header và phần hiển thị profile

Tùy ngữ cảnh, tùy mục đích các layout sẽ được sử dụng cho phù hợp.

- + **components (thành phần)**: chứa các component riêng lẻ hoạt động độc lập

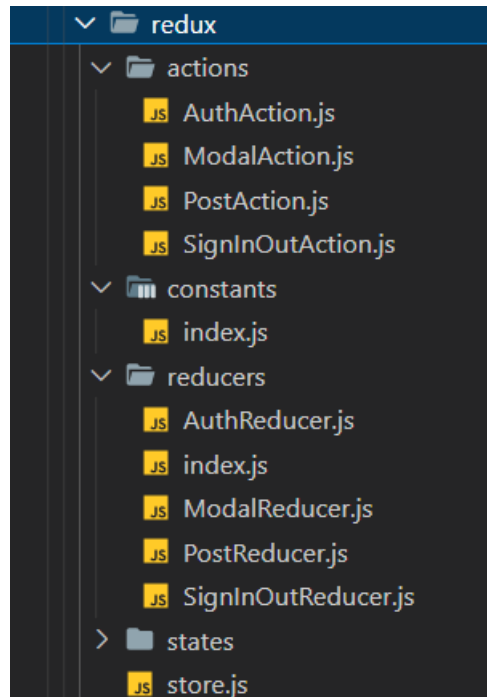


Hình 8: Cấu trúc thành phần components

Component trong ReactJS hay các thành phần là các cấu trúc cốt lõi của React. Nói cách khác, mọi ứng dụng được phát triển bởi React sẽ được tạo thành từ các thành phần được gọi là Component. Các Component này làm việc xây dựng UIs trở nên dễ dàng hơn rất nhiều. Giao diện người dùng được chia thành nhiều phần riêng lẻ được gọi là các component và hoạt động độc lập. Khi hợp nhất tất cả các component riêng lẻ này, ta sẽ nhận được giao diện người dùng cuối cùng. Ví dụ, Topbar (hay Header) là một thành phần được gọi bởi DefaultLayout, mặc khác DefaultLayout cũng là một thành phần bởi nó có thể được gọi tên bởi các thành phần khác khi cần thiết; thậm chí, một Button cũng là một component.

Ưu điểm lớn nhất là các component này có thể tái sử dụng nhiều lần, thuận tiện cho công việc chỉnh sửa, debug.

- + **Redux**: công cụ quản lý state được sử dụng trong ứng dụng React

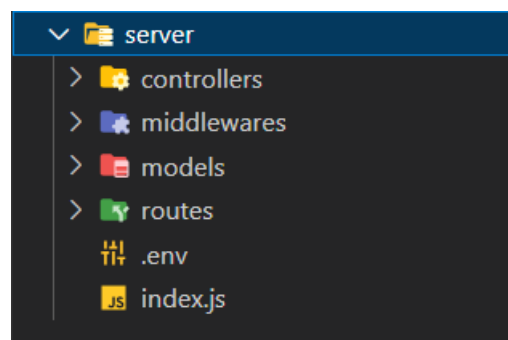


Hình 9: Cấu trúc thành phần Redux

Mặc dù được sử dụng chủ yếu với React, nhưng redux có thể được sử dụng với bất kỳ khung hoặc thư viện JavaScript nào khác. Với Redux, trạng thái hệ thống được giữ trong một "store" và mỗi thành phần có thể truy cập bất kỳ trạng thái nào mà nó cần từ "store" này. Một "store" có thể xây dựng gồm 3 thành phần: actions, store và reducers

Cấu trúc phía server

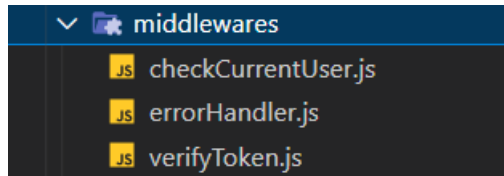
Server được xây dựng theo mô hình MVC giúp tổ chức code theo từng phần độc lập với nhau và các phần tương tác với nhau theo một cách nhất định. Có thể xây dựng gồm 3 thành phần chính: model, controller, route (view).



Hình 10: Cấu trúc server

Middlewares: các thành phần trung gian xử lý các sự kiện nói chung nó chính là các function có thể tái sử dụng nhiều lần.

Ví dụ như kiểm tra người dùng đã đăng nhập hay chưa, xác thực token để đảm bảo tính bảo mật hay là dùng để xử lý các error.



Hình 11: Cấu trúc thành phần middlewares

.env: file chứa các biến môi trường

3.4. Thiết kế cấu trúc dữ liệu (Model)

Hệ thống có 2 đối tượng chính đó là post và user được xây dựng như sau:

- Post
 - Post-id
 - Content (nội dung của bài post)
 - createdAt (ngày tạo)
 - author (tác giả)
 - tag (thẻ truy xuất của post)
 - favorite (lượt yêu thích)
 - comment (lượt bình luận)
 - shared (lượt chia sẻ)
- User
 - User-id
 - Username (tên người dùng)
 - Email (tài khoản email)
 - Password (mật khẩu)
 - Policy (true/false yêu cầu điều khoản)
 - Avatar
 - Background
 - firstName
 - lastName

- signature
- following (danh sách người đang theo dõi)
- follower (danh sách người theo dõi)

3.5. Thiết kế API chuẩn RESTful (Route)

Đây chính là công đoạn xây dựng Route – công cho phép truy cập tài nguyên, dữ liệu

auth

- server/auth/ - GET (lấy thông tin user)
- server/auth/register – POST (đăng ký)
- server/auth/login – POST (đăng nhập)

Code preview:

```
// POST - server/auth/register
router.post("/register", register);

// POST - server/auth/login
router.post("/login", login);

// POST - server/auth/
router.route('/').get(checkCurrentUser, getCurrentUser)
```

Hình 12: Code preview authRoute

post

- server/post/ - GET (lấy toàn bộ post) – POST (tạo một post)
- server/post/:postId – PUT (cập nhật post) – DELETE (xóa một post)

Code preview:

```
router.route('/').get(getAllPosts).post(verifyToken, createAPost)

router.route('/:postId').put(verifyToken, updateAPost).delete(verifyToken, deleteAPost)
```

Hình 13: Code preview postRoute

user

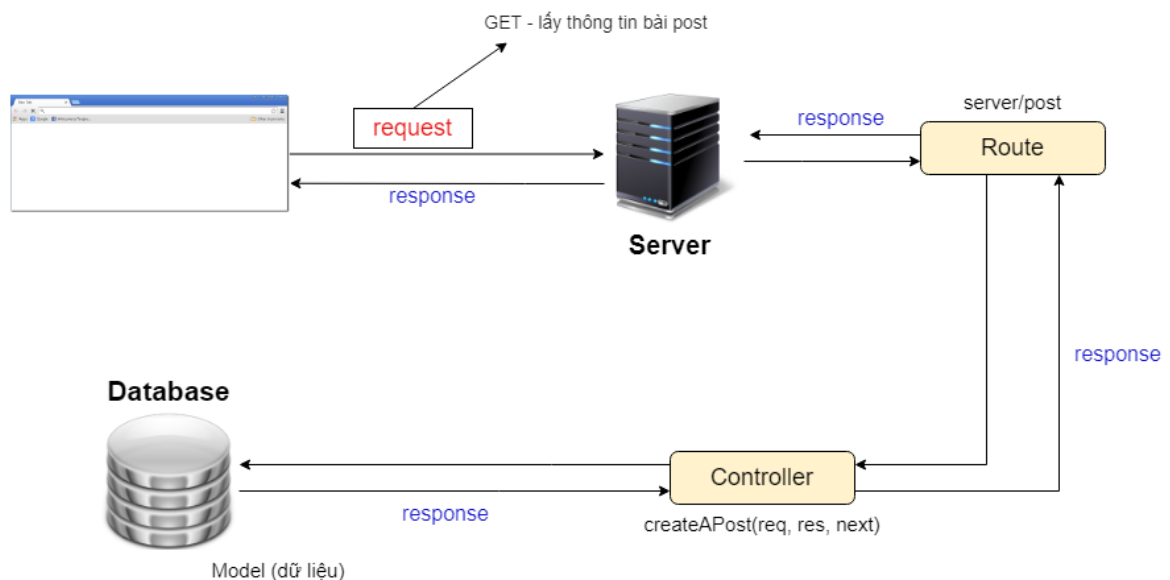
- server/user/:username – GET (lấy thông tin user) – PUT (cập nhật thông tin user) – DELETE (xóa một user)
- server/user/:username/follow – PUT (follow và unfollow)

Code preview:

```
router.route("/:username").get(verifyToken, getUser).put(verifyToken, updateUser).delete(verifyToken, deleteUser)
router.route("/:username/follow").put(verifyToken, followUser)
```

Hình 14: Code preview userRoute

3.6 Thiết kế Controller tương ứng với từng API



Hình 15: Quy trình làm việc của Back-end

Quy trình làm việc của Back-end

- Client gửi một request đến server
- Server truy tìm các route có phương thức GET tương ứng với request
- Route với phương thức GET sẽ tới controller, controller tiếp nhận xử lý request
- Controller gồm các function sẽ xử lý và kết nối với database lấy những dữ liệu cần thiết
- Cuối cùng phản hồi và hiển thị ra màn hình

Vì thế các controller được xây tương ứng với các route như sau:

- Tương ứng với authRoute ta có authController:

```
// GET CURRENT USER
exports.getCurrentUser = async (req, res, next) => {
  try {
    const data = { user: null }
    if (req.user) {
      const user = await User.findOne({ _id: req.user.userId })
      data.user = user
    }
    res.status(200).json({
      status: 'success',
      data: data
    })
  } catch (error) {
    res.json(error)
  }
}
```

Hình 16: Code preview Get Current User trong authController

```
// LOGIN CONTROLLER
exports.login = async (req, res, next) => {
  try {
    const user = await User.findOne({ username: req.body.username })

    // Error: Email is not correct
    if (!user) {
      const err = new Error('Tài khoản/email chưa được đăng ký')
      err.statusCode = 400
      return next(err)
    }

    // Check password
    if (bcrypt.compareSync(req.body.password, user.password)) {
      const token = jwt.sign({ userId: user._id }, process.env.APP_SECRET)
      res.status(200).json({
        status: 'success',
        data: {
          message: "Đăng nhập thành công",
          token,
          userId: user._id,
        }
      })
    } else {
      // Error: Password is not correct
      const err = new Error("Sai mật khẩu")
      err.statusCode = 400
      return next(err)
    }
  } catch (error) {
    res.json(error)
  }
}
```

Hình 17: Code preview LoginController trong authController

```
// REGISTER CONTROLLER
exports.register = async (req, res, next) => {

  try {
    const { username, password, confirmPassword, policy } = req.
body

    // Check username is not empty
    if (username === "" || password === "" || confirmPassword ===
    "") {
      const err = new Error("Vui lòng nhập đầy đủ thông tin")
      err.statusCode = 400
      return next(err)
    }

    // Check password duplication
    if (password !== confirmPassword) {
      const err = new Error("Mật khẩu không trùng nhau")
      err.statusCode = 400
      return next(err)
    }

    // Check policy
    if (policy === false) {
      const err = new Error(
        "Bạn có đồng ý với chính sách của chúng tôi")
      err.statusCode = 400
      return next(err)
    }

    const user = await User.create(req.body)
    const token = jwt.sign({ userId: user._id }, process.env.
APP_SECRET)

    res.status(200).json({
      status: 'success',
      data: { token, userId: user._id },
      message: "Đăng ký thành công"
    })
  } catch (error) {
    next(error)
  }
}
```

Hình 18: Code preview RegisterContronller trong authContronller

- Tương ứng với postRoute ta có các controller như sau:

```
// GET ALL POSTS
exports.getAllPosts = async (req, res, next) => {
  try {
    const posts = await Post.find({}).populate('author',
'username avatar').select('content createdAt updatedAt')
    res.status(200).json({
      status: 'success',
      result: posts.length,
      data: { posts }
    })
  } catch (error) {
    res.json(error)
  }
}
```

Hình 19: Code preview getAllPost contronller

```
// GET A POSTS
exports.getAPost = async (req, res, next) => {
  try {
    const posts = await Post.find({}).populate('author')
    res.status(200).json({
      status: 'success',
      result: posts.length,
      data: { posts }
    })
  } catch (error) {
    res.json(error)
  }
}
```

Hình 20: Code preview getAPost contronller

```
// CREATE A POST
exports.createAPost = async (req, res, next) => {
  try {
    const { userId } = req.user;

    const post = await Post.create({ ...req.body, author: userId })
    res.status(200).json({
      status: 'success',
      data: { post }
    })
  } catch (error) {
    next(error)
  }
}
```

Hình 21: Code preview createAPost contronller

```
// UPDATE A POST
exports.updateAPost = async (req, res, next) => {
  try {
    const { postId } = req.params;

    const post = await Post.findByIdAndUpdate(postId, { ...req.body }, { new: true, runValidators: true })

    res.status(200).json({
      status: 'success',
      data: { post }
    })
  } catch (error) {
    next(error)
  }
}
```

Hình 22: Code preview updateAPost contronller

```
// DELETE A POST
exports.deleteAPost = async (req, res, next) => {
  try {
    const { postId } = req.params;

    await Post.findByIdAndDelete(postId)

    res.status(200).json({
      status: 'success',
      message: 'Post has been deleted'
    })
  } catch (error) {
    next(error)
  }
}
```

Hình 23: Code preview deleteAPost contronller

- Tương ứng với userRoute ta có các controller:

```
// Follow a user
exports.followAUser = async (req, res, next) => {
  try {
    if (req.body.currentUsername !== req.params.username) {

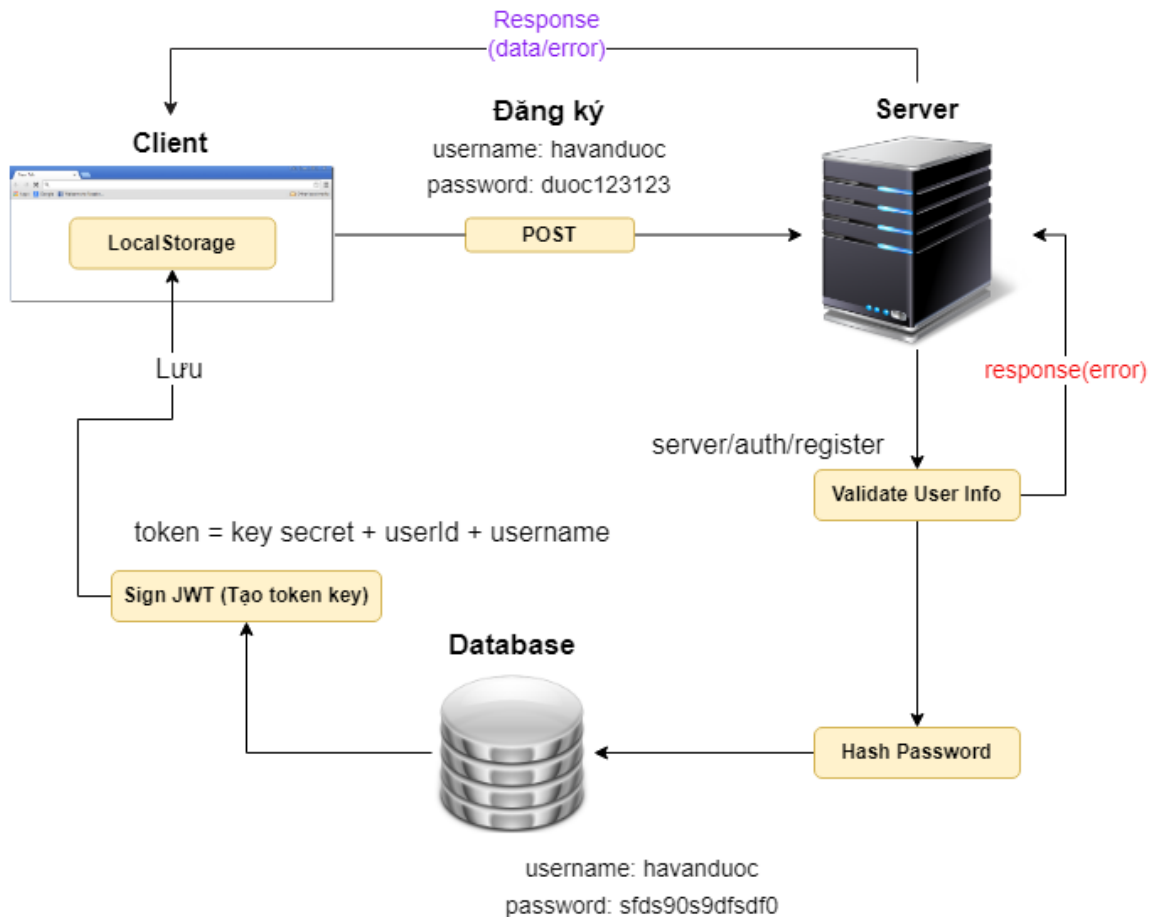
      currentUser = await User.findOne({ username: req.body.currentUsername });
      userIsFollowed = await User.findOne({ username: req.params.username });

      if (!currentUser.following.includes(req.params.username)) {
        await currentUser.updateOne({ $push: { following: req.params.username } })
        await userIsFollowed.updateOne({ $push: { follower: req.body.currentUsername } })
        res.json({
          status: "success",
          message: `Bạn đã thành công theo dõi người dùng ${req.body.currentUsername}`
        })
      } else {
        res.json({
          status: "success",
          message: `Bạn đã theo dõi người dùng ${req.body.currentUsername} này rồi`
        })
      }
    } else {
      res.json("Bạn không thể theo dõi chính mình");
    }
  } catch (error) {
    res.json(error)
  }
}
```

Hình 24: Code preview Follow A User contronller

3.7 Quy trình đăng ký, đăng nhập

REGISTER WITH JWT AUTHENTICATION WORKFLOW



Hình 25: Quy trình đăng ký

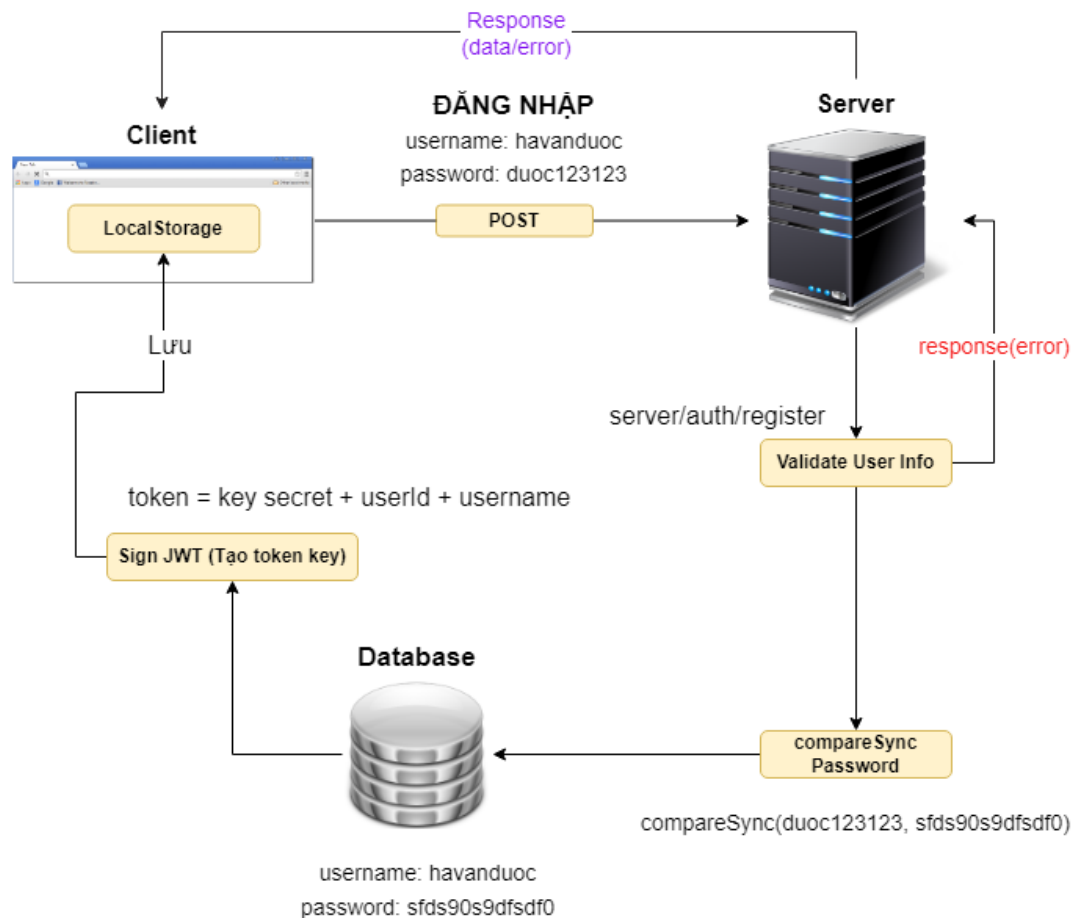
Các bước trong quy trình:

- Client gửi request đến server ở đây là POST Đăng ký hoặc đăng nhập gồm các thông tin username, password...
- Server sẽ xác thực thông tin như kiểm tra tài khoản đã tồn tại hay chưa, mật khẩu đã đúng yêu cầu hay chưa. Trường hợp xảy ra lỗi sẽ phản hồi về server.
- Sau đó hash password (mã hóa password) lưu vào database. Đồng thời đăng ký một token key – token key là một đoạn mã hóa bao gồm userId, username và một ký tự bí mật do người lập trình quy định.
- Cuối cùng token key này sẽ được lưu vào LocalStorage của trình duyệt.

3.8 Quy trình đăng nhập

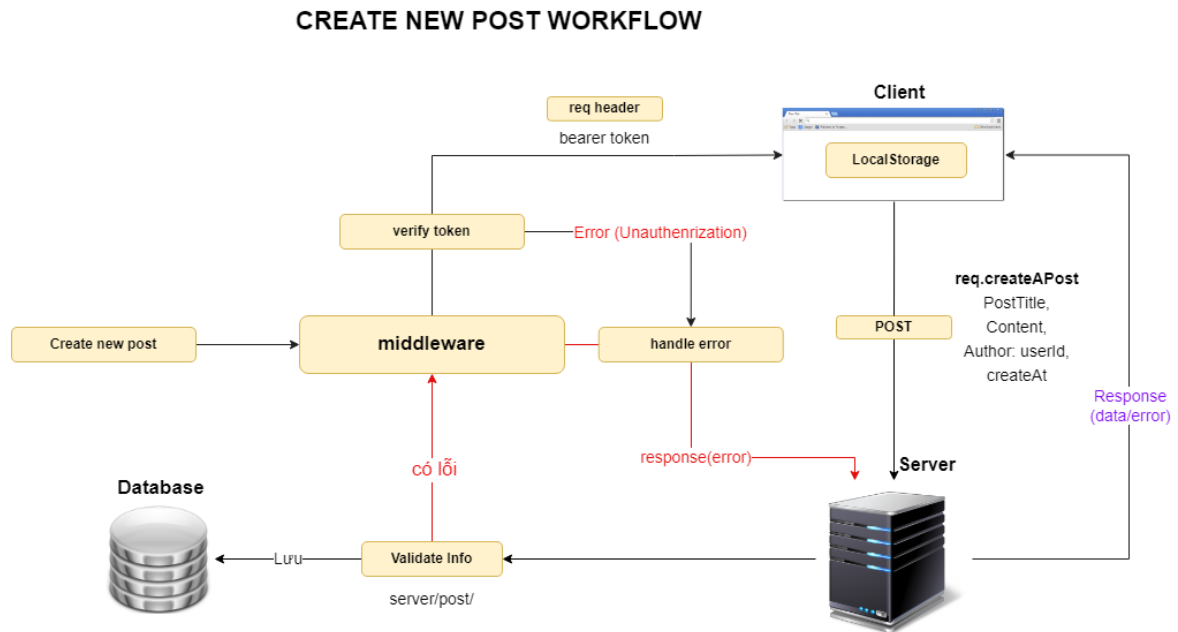
Tương tự với quy trình đăng ký tuy nhiên ta sẽ không cần hash password mà sẽ compareSync password – so sánh mật khẩu nhập vào và mật khẩu đã mã hóa được lưu trong database.

LOGIN WITH JWT AUTHENTICATION WORKFLOW



Hình 26: Quy trình đăng nhập

3.8 Quy trình Tạo/Chỉnh sửa/Xóa/Lấy post



Hình 27: Quy trình Tạo bài post

Quy trình tuần tự theo các bước:

- Khi tạo một bài post đầu tiên phải xác thực token (mỗi user có một token không trùng nhau). Nếu không phải là user đã đăng nhập sẽ phản hồi Error Unauthenrization.
- Sau khi xác thực token thành công client gửi request gồm các thông tin bài post đến server xử lý và sau cùng lưu vào database. Trường hợp xảy ra lỗi sẽ xử lý lỗi và phản hồi về server.

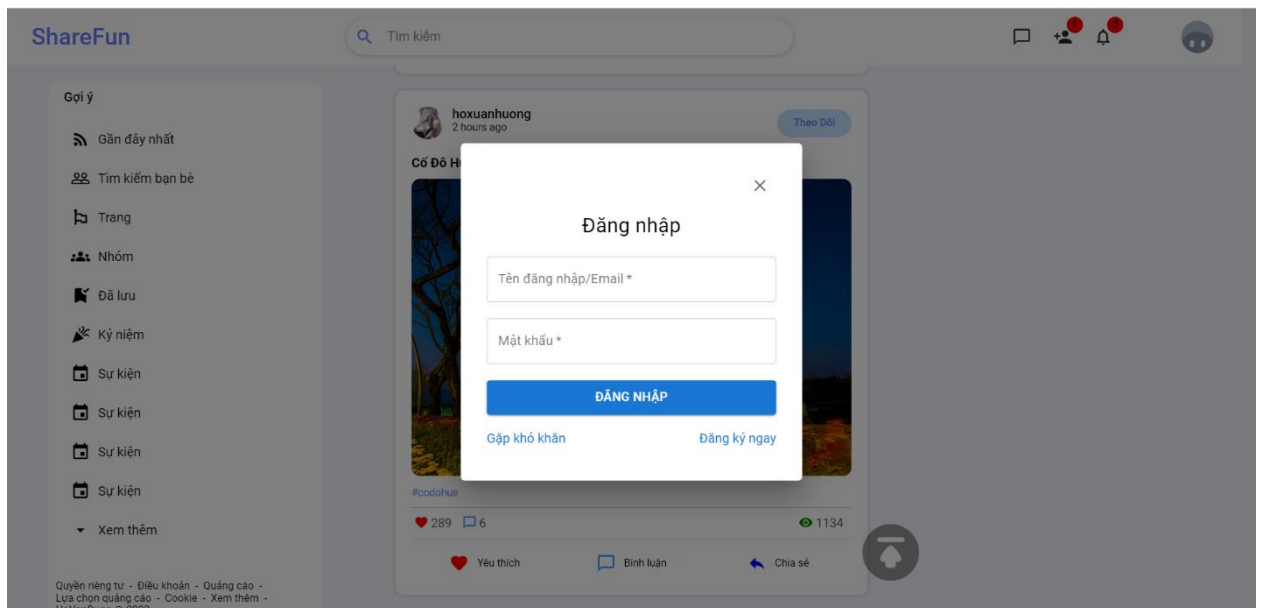
Các quy trình chỉnh sửa, xóa, lấy post tương tự như quy trình tạo post. Điểm khác biệt ở request với chỉnh sửa đó là phương thức PUT, xóa là phương thức DELETE và lấy post là phương thức GET.

Chương 4: THIẾT KẾ GIAO DIỆN WEBSITE

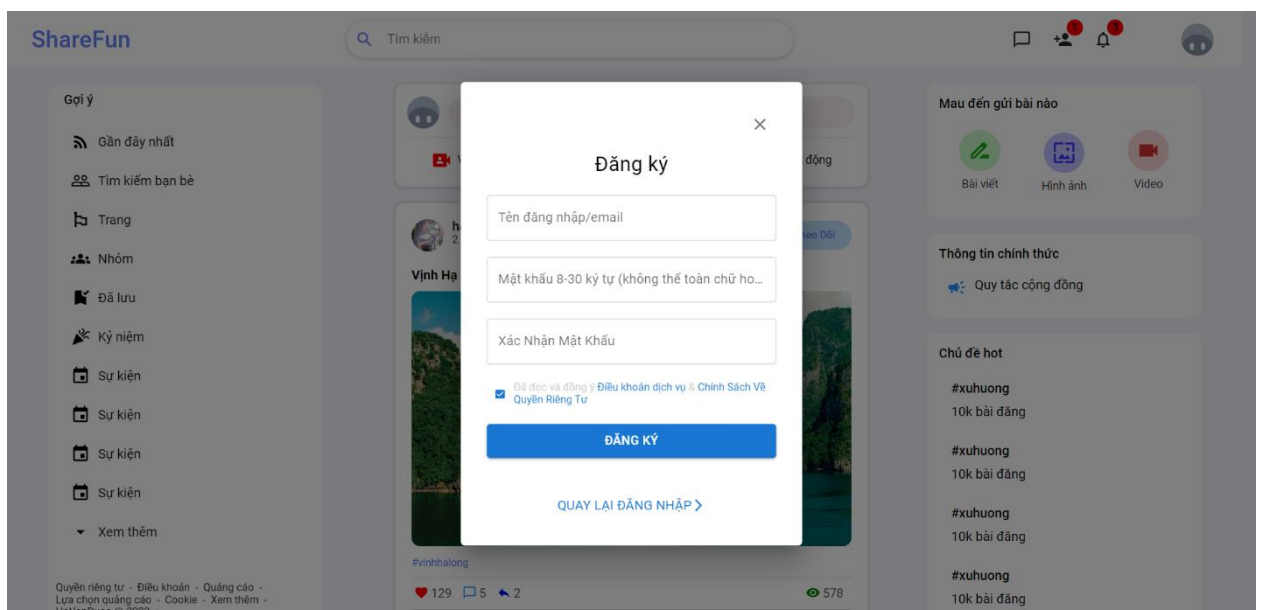
4.1. Modal đăng nhập, đăng ký

Khi click chuột tương tác hệ thống sẽ kiểm tra LocalStorage đã tồn tại token key hay chưa. Nếu chưa sẽ hiển thị modal Login yêu cầu đăng nhập.

Sau khi đăng nhập thành công thông tin user sẽ được mã hóa dưới dạng token key và lưu trong LocalStorage của trình duyệt.

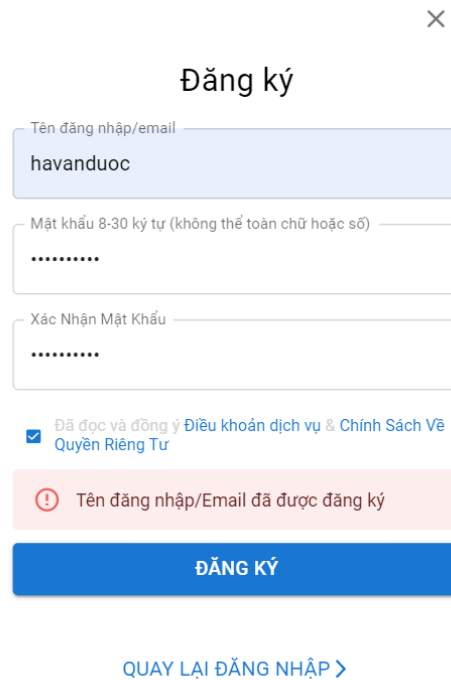


Hình 28: Giao diện Modal Login



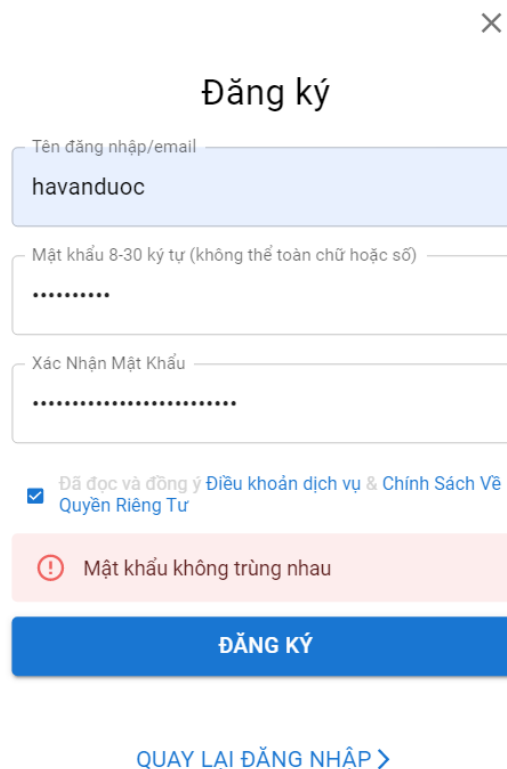
Hình 29: Giao diện Modal đăng ký

Trường hợp xảy ra lỗi server sẽ phản hồi lỗi và hiển thị ra màn hình như sau:



The screenshot shows a registration form titled "Đăng ký" (Register). It includes fields for "Tên đăng nhập/email" (Username/email) with the value "havanduoc", "Mật khẩu 8-30 ký tự (không thể toàn chữ hoặc số)" (Password 8-30 characters, cannot be all letters or numbers), and "Xác Nhận Mật Khẩu" (Confirm Password). A checkbox for "Đã đọc và đồng ý Điều khoản dịch vụ & Chính Sách Về Quyền Riêng Tư" (I have read and agree to the Terms of Service & Privacy Policy) is checked. A red error message box states: "Tên đăng nhập/Email đã được đăng ký" (Username/Email already registered). Below the error message is a blue "ĐĂNG KÝ" (REGISTER) button and a link "QUAY LẠI ĐĂNG NHẬP >" (GO BACK TO LOGIN >).

Hình 30: Lỗi đăng ký "Tên đăng nhập đã tồn tại"



The screenshot shows the same registration form as Figure 30. The "Tên đăng nhập/email" field still contains "havanduoc". The "Mật khẩu" field now contains "....." and the "Xác Nhận Mật Khẩu" field contains ".....". The red error message box states: "Mật khẩu không trùng nhau" (Passwords do not match). The "ĐĂNG KÝ" button and the "QUAY LẠI ĐĂNG NHẬP >" link are still present.

Hình 31: Lỗi đăng ký "Mật khẩu không trùng nhau"

×

Đăng nhập

Tên đăng nhập/Email *

Mật khẩu *

!
Sai mật khẩu

ĐĂNG NHẬP

[Gặp khó khăn](#)
[Đăng ký ngay](#)

Hình 32: Lỗi đăng nhập "Sai mật khẩu"

×

Đăng nhập

Tên đăng nhập/Email *

Mật khẩu *

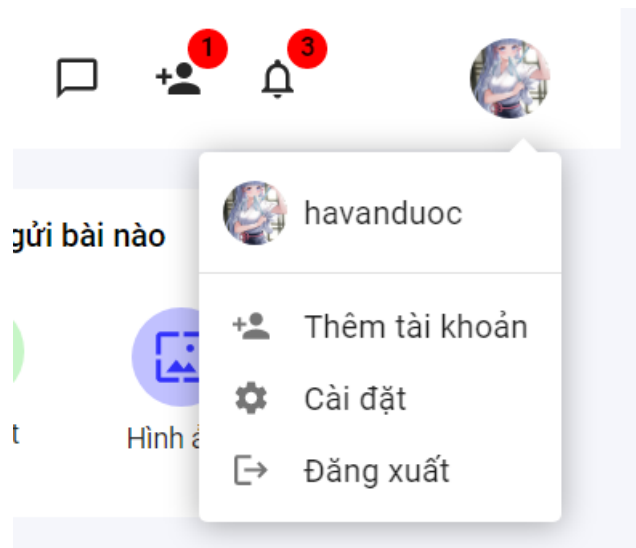
!
Tài khoản/email chưa được đăng ký

ĐĂNG NHẬP

[Gặp khó khăn](#)
[Đăng ký ngay](#)

Hình 33: Lỗi đăng nhập "Tài khoản/ email chưa được đăng ký"

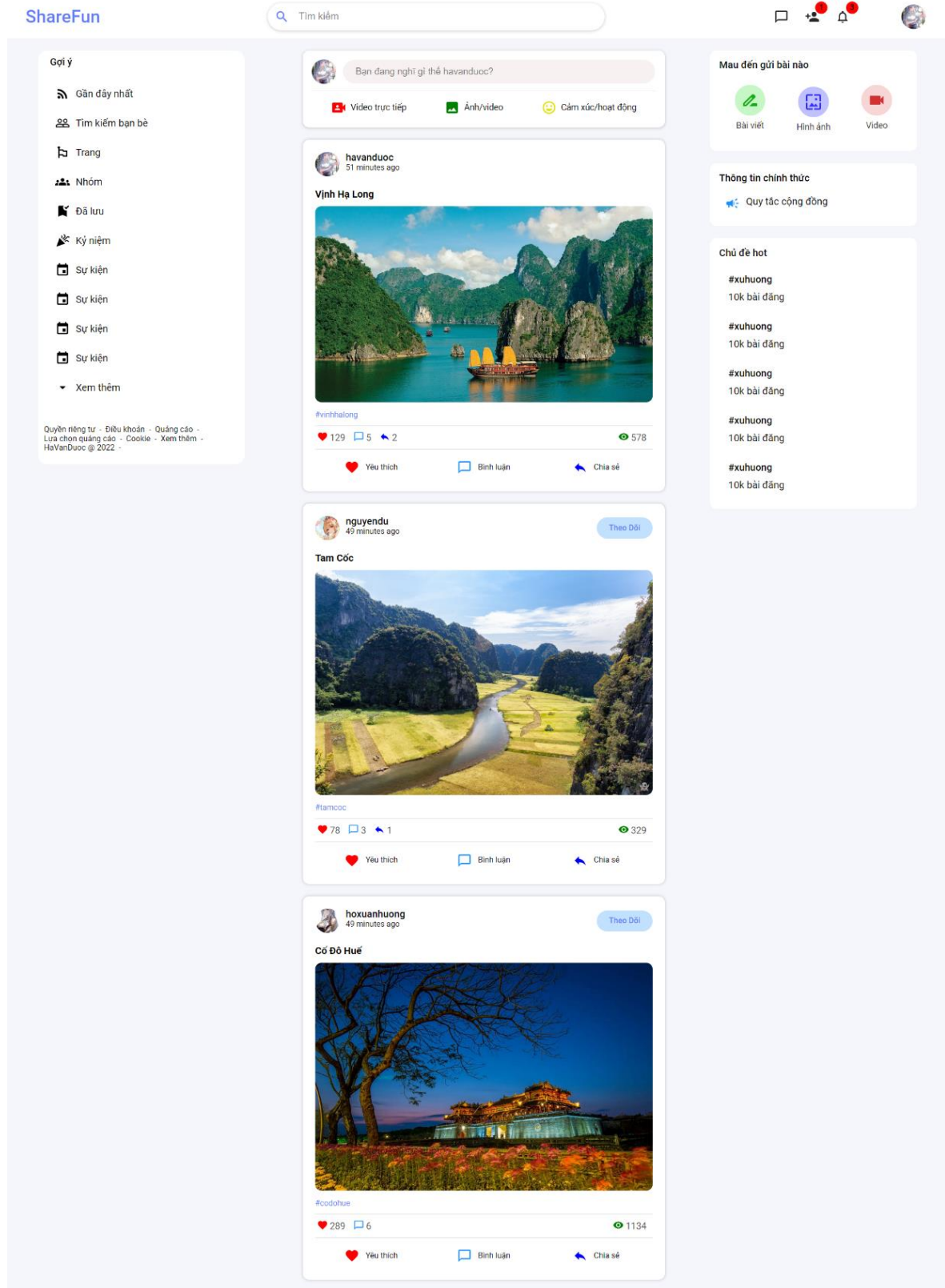
Để đăng xuất tài khoản, ta nhấn chọn button account bên phải của thanh header sẽ hiển thị popup và chọn Đăng xuất. Sau khi chọn đăng xuất hệ thống sẽ xóa toàn bộ token được lưu trong LocalStorage.



Hình 34: Giao diện tính năng đăng xuất

4.2. Trang timeline (trang chủ)

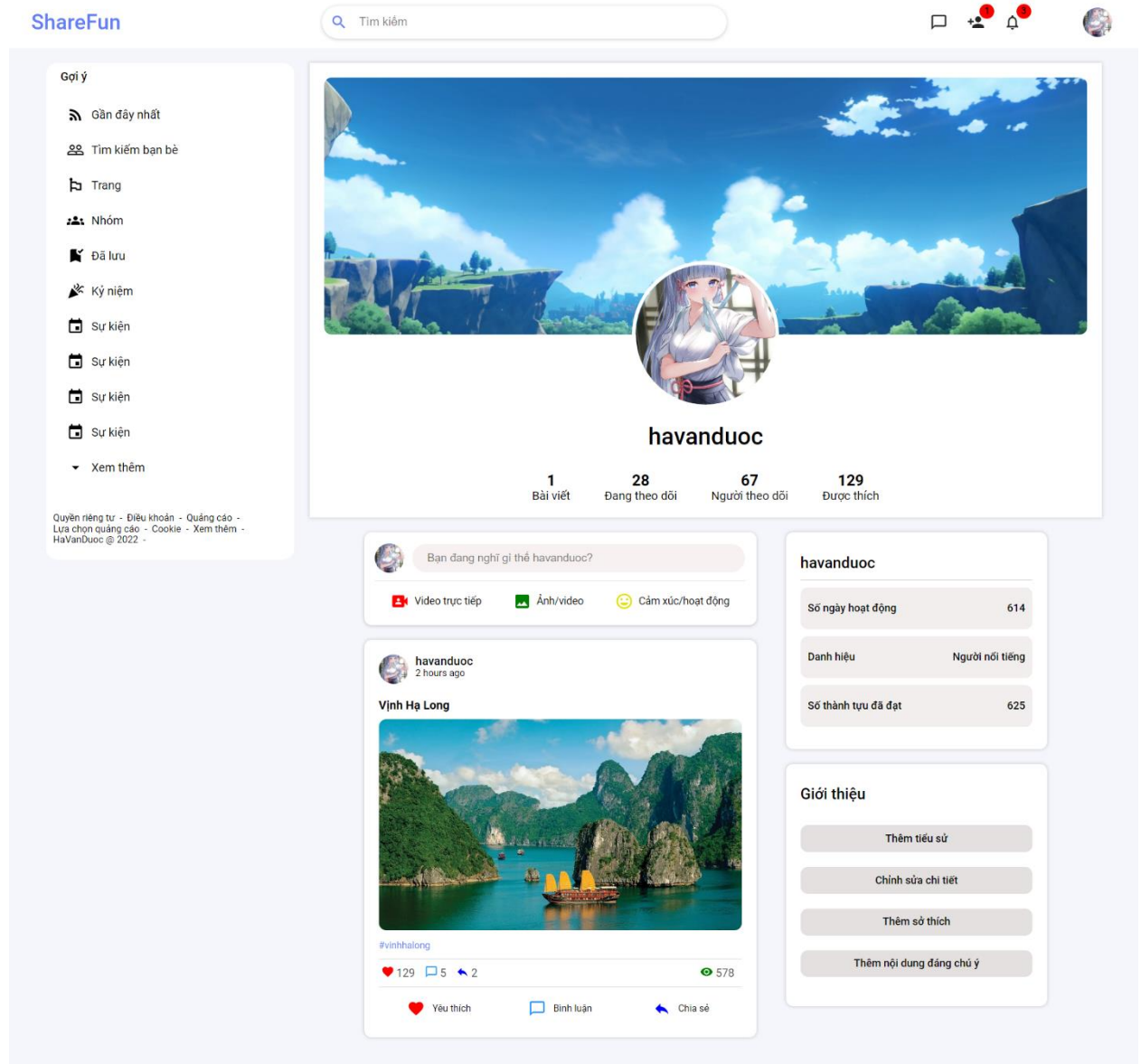
- Trang chủ hiển thị tất cả các bài post liên quan.
- Các tính năng trên một bài post như follow, like, comment, share...



Hình 35: Giao diện Trang chủ

4.3. Trang profile (hồ sơ cá nhân)

Trang profile sẽ hiển thị toàn bộ thông tin của người dùng đã đăng nhập



Hình 36: Giao diện Trang profile

4.4. Trang tạo bài post

Để tạo một bài post yêu cầu các nội dung:

- Tiêu đề
- Nội dung, hình ảnh
- Chủ đề (tag)

Hình 37: Giao diện Trang tạo bài post

Chương 5: TỔNG KẾT THÀNH TỰU VÀ HƯỚNG PHÁT TRIỂN

5.1. Thành tựu đạt được

- + Tiếp cận thực tế, quy trình làm việc tại nơi thực tập.
- + Xây dựng được các mối quan hệ tốt đẹp với đồng nghiệp nơi thực tập.
- + Xây dựng hoàn thành hệ thống website với các tính năng cơ bản như đăng nhập, đăng ký, trình bày các bài post, tạo bài post ...
- + Biết thêm nhiều công nghệ thịnh hành như React, Express, MVC...
- + Hoàn thành bài báo cáo đúng yêu cầu, đúng thời hạn.

5.2. Hạn chế

- + Do thời gian hạn chế nên vẫn còn chức năng còn trên ý tưởng chưa thực hiện trực tiếp vào webiste.
- + Website mới chỉ thực thi được ở quy mô nhỏ, hẹp, chưa mở rộng sự quản lý của đề tài.
- + Giao diện người dùng chưa được thực sự bắt mắt, thu hút người dùng.
- + Một số vấn đề về cơ sở dữ liệu chưa xử lý được.

5.3. Hướng phát triển

- + Tìm hiểu và áp dụng một số chức năng khác như box chat, call video...
- + Mở rộng quy mô hệ thống