**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI VIỆN CÔNG NGHỆ THÔNG TIN, KHOA CÔNG NGHỆ PHẦN MỀM**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**XÂY DỰNG WEBSITE CHIA SẺ KINH NGHIỆM DU LỊCH**

**PHẠM THỊ THẢO**

**Hà Nội , 7/2022**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI VIỆN CÔNG NGHỆ THÔNG TIN, KHOA CÔNG NGHỆ PHẦN MỀM**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**XÂY DỰNG WEBSITE CHIA SẺ KINH NGHIỆM DU LỊCH**

Giáo viên hướng dẫn **: Nguyễn Tiến Thành**

Sinh viên **: Phạm Thị Thảo**

**Hà Nội , 7/2022**

Lời cam kết

Họ và tên sinh viên: Phạm Thị Thảo

Điện thoại liên lạc: 0355678384 Email: [thao.pt153455@sis.hust.edu.vn](mailto:thao.pt153455@sis.hust.edu.vn)

Lớp: CNTT 2-2 K60 Hệ đào tạo: Chính Quy

Tôi Phạm Thị Thảo cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của TS Nguyễn Tiến Thành. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

*Hà Nội, ngày 12 tháng 06 năm 2022*

Tác giả ĐATN

*Phạm Thị Thảo*

# Lời cảm ơn

Cuộc đời của mỗi người đều là một cuộc hành trình mà ở đó mỗi chúng ta đều cần phải không ngừng học hỏi, trau dồi thêm tri thức của bản thân mình. Bản thân em vô cùng biết ơn cha mẹ, những người mang em đến với thế giới này, nuôi em khôn lớn, tần tảo sớm hôm để em có được ngày hôm nay. Em luôn tự hào khi được giới thiệu với tất cả mọi người “Mình là sinh viên Bách Khoa” – nơi giúp em mở mang kiến thức, được học hỏi để dần hoàn thiện chính bản thân, nhưng Bách Khoa cũng lưu giữ lại những kỷ niệm đáng nhớ, những khoảnh khắc và nơi đặt dấu mốc cho sự trưởng thành của em. Em xin gửi lời cảm ơn chân thành nhất đến nhà trường và những thầy cô trong trường Đại học Bách Khoa Hà Nội- những người tận tình dạy dỗ mang đến tri thức và bài học về cuộc đời, bài học về cách làm người, đó là những hành trang giúp em ngày càng vững bước trên quãng đường sau này.

Em cũng xin gửi lời cảm ơn sâu sắc tới TS. Nguyễn Tiến Thành, em cảm ơn thầy đã luôn nhiệt tình, hướng dẫn và chỉ dạy em sát sao trong quá trình em làm đồ án, những lời khuyên và đánh giá của thầy đã giúp em hoàn thiện đồ án của mình tốt hơn và có cái nhìn nghiêm túc về sản phẩm của một kỹ sư.

Em cũng xin bày tỏ lời cảm ơn đến những người bạn đại học đã luôn sẵn sàng giúp đỡ, chia sẻ và động viên em trong những khoảng thời gian khó khăn

# Tóm tắt

Hiện nay xã hội ngày càng phát triển đối với tất cả các ngành nghề, đặc biệt là du lịch. Xã hội phát triển đi đôi với việc nhu cầu cuộc sống cũng tăng cao, theo đó nhu cầu về du lịch cũng cần thiết đối với mọi người. Đối với mọi người nói chung và giới trẻ nói riêng du lịch đang là các hoạt động không thể thiếu trong cuộc sống vì nó mang tính giải trí, trải nghiệm rất cao. Trên thực tế rất nhiều những trang web, những blogger chia sẻ kinh nghiệm du lịch nhưng đa số theo xu hướng Marketing, quảng cáo dịch vụ khiến người dùng cảm thấy nhàm chán và không thiết thực.

Nhận ra những vấn đề về việc chia sẻ kinh nghiệm du lịch của người dùng nên em quyết định xây dựng một website để phục vụ cho mọi người dễ dàng tìm kiếm và cập nhật được thị hiếu của điểm đến mà người dùng muốn đến. Website hướng đến sự thân thiện với người dùng và những review thực tế nhất để mọi người có kinh nghiệm cũng như sự an tâm khi lựa chọn địa điểm phù hợp với mình muốn đến. Ứng dụng được xây dựng trên nền tang website để người dùng có thể dễ dàng truy cập. Đặc điểm để tạo nên website chính là thân thiện với người dùng và đem lại những review chân thực nhất.

Mục lục

[Lời cam kết 1](#_Toc107433830)

[Lời cảm ơn 2](#_Toc107433831)

[Tóm tắt 3](#_Toc107433832)

[Mục lục 4](#_Toc107433833)

[Danh mục hình vẽ 7](#_Toc107433834)

[Danh mục bảng 8](#_Toc107433835)

[Danh mục các từ viết tắt 9](#_Toc107433836)

[Danh mục thuật ngữ 10](#_Toc107433837)

[Chương 1: Giới thiệu đề tài 11](#_Toc107433838)

[1.1. Đặt vấn đề 11](#_Toc107433839)

[1.2. Mục tiêu và phạm vi của đề tài 11](#_Toc107433840)

[1.3. Định hướng giải pháp 12](#_Toc107433841)

[1.4. Bố cục của đồ án 13](#_Toc107433842)

[Chương 2: Khảo sát và phân tích yêu cầu 14](#_Toc107433843)

[2.1 Khảo sát hiện trạng 14](#_Toc107433844)

[2.2 Tổng quan chức năng 14](#_Toc107433845)

[*2.2.1* *Biểu đồ use case tổng quan* 14](#_Toc107433846)

[*2.2.2* *Biểu đồ use case phân rã* 15](#_Toc107433847)

[2.3 Đặc tả chức năng 19](#_Toc107433848)

[*2.3.1* *Đặc tả use case đăng nhập, đăng ký* 19](#_Toc107433849)

[*2.3.2* *Đặc tả use case tương tác với bài viết* 22](#_Toc107433850)

[*2.3.3* *Đặc tả use case tạo mới bài viết* 23](#_Toc107433851)

[*2.3.4* *Đặc tả use case quản lý bài viết của Admin* 24](#_Toc107433852)

[2.4 Yêu cầu phi chức năng (Bảo mật của API) 26](#_Toc107433853)

[Chương 3: Công nghệ sử dụng 28](#_Toc107433854)

[3.1 ReactJS 28](#_Toc107433855)

[*3.1.1* *Khái niệm về ReactJS* 28](#_Toc107433856)

[*3.1.2* *Đặc trưng của ReactJS* 28](#_Toc107433857)

[3.2 Redux 34](#_Toc107433858)

[*3.2.1* *Giới thiệu về Redux* 34](#_Toc107433859)

[*3.2.2* *Redux và ReactJS* 34](#_Toc107433860)

[*3.2.3* *Hoạt động của Redux* 34](#_Toc107433861)

[3.3 MongoDB 35](#_Toc107433862)

[*3.3.1* *Giới thiệu về MongoDB* 35](#_Toc107433863)

[*3.3.2* *Ưu nhược điểm của MongoDB* 37](#_Toc107433864)

[3.4 NodeJS 38](#_Toc107433865)

[*3.4.1* *Giới thiệu về NodeJS* 38](#_Toc107433866)

[*3.4.2* *NodeJS hoạt động như thế nào?* 38](#_Toc107433867)

[*3.4.3* *MongooseJS* 39](#_Toc107433868)

[*3.5* *Tailwind CSS* 39](#_Toc107433869)

[*3.5.1* *Tailwind CSS là gì ?* 39](#_Toc107433870)

[*3.5.2* *Tại sao chúng ra nên dùng Tailwind?* 39](#_Toc107433871)

[3.6 Bảo mật API với Json web token – JWT 39](#_Toc107433872)

[*3.5.1* *JSON Web Token là gì?* 39](#_Toc107433873)

[*3.5.2* *Khi nào nên sử dụng JSON Web Token?* 40](#_Toc107433874)

[*3.5.3* *Cấu trúc của Json Web Token* 40](#_Toc107433875)

[Chương 4: Phát triển và triển khai ứng dụng 43](#_Toc107433876)

[4.1 Thiết kế kiến trúc 43](#_Toc107433877)

[*4.1.1* *Biểu đồ tuần tự toàn hệ thống* 43](#_Toc107433878)

[*4.1.2* *Biểu đồ tuần tự UC Đăng nhập* 43](#_Toc107433879)

[*4.1.3* *Biểu đồ tuần tự UC Đăng ký* 44](#_Toc107433880)

[4.2 Mô hình hóa dữ liệu 45](#_Toc107433881)

[*4.2.1* *Xác định thực thể* 45](#_Toc107433882)

[*4.2.2* *Xác định liên kết giữa các thực thể* 46](#_Toc107433883)

[*4.2.3* *Thiết kế cơ sở dữ liệu MongoDB* 47](#_Toc107433884)

[4.3 Xây dựng ứng dụng 50](#_Toc107433885)

[*4.4.1* *Thư viện và công dụng sử dụng* 50](#_Toc107433886)

[*4.4.2* *Kết quả đạt được* 51](#_Toc107433887)

[*4.4.3* *Minh họa các chức năng chính* 51](#_Toc107433888)

[Chương 5: Các giải pháp và đóng góp nổi bật 59](#_Toc107433889)

[5.1 Cải thiện cấu trúc Font-end để tăng tốc website 59](#_Toc107433890)

[5.2 Chức năng tạo bài viết chia sẻ kinh nghiệm du lịch 60](#_Toc107433891)

[*5.2.1* *Giới thiệu chức năng* 60](#_Toc107433892)

[*5.2.2* *Các hướng tiếp cận* 60](#_Toc107433893)

[*5.2.3* *Giải pháp* 61](#_Toc107433894)

[*5.2.4* *Kết quả thu được* 61](#_Toc107433895)

[Chương 6: Kết luận và hương phát triển 63](#_Toc107433896)

[6.1 Kết luận 63](#_Toc107433897)

[6.2 Hướng phát triển 63](#_Toc107433898)

[Tài liệu tham khảo 65](#_Toc107433899)

**Danh mục hình vẽ**

[**Hình 1** Biểu đồ use case tổng quan. 13](#_Toc107268916)

[**Hình 2** Biểu đồ phân rã use case đăng nhập, đăng ký. 14](#_Toc107268917)

[**Hình 3** Biểu đồ phân rã use case tương tác với bài viết. 15](#_Toc107268918)

[**Hình 4** Biểu đồ phân rã use case tạo mới bài viết. 16](#_Toc107268919)

[**Hình 5** Biểu đồ phân rã use case quản lý bài viết của Admin. 17](#_Toc107268920)

[**Hình 6** Cấu trúc HTML sử dụng JSX. 23](#_Toc107268921)

[**Hình 7** Cấu trúc HTML DOM. 24](#_Toc107268922)

[**Hình 8** Quá trình thực hiện server side rendering. 26](#_Toc107268923)

[**Hình 9** Quá trình thực hiện client side rendering. 27](#_Toc107268924)

[**Hình 10** Quản lý dữ liệu có redux và không có redux [9]. 29](#_Toc107268925)

[**Hình 11** Màn hình trang chủ khi chưa đăng nhập. 38](#_Toc107268926)

[**Hình 12** Màn hình trang chủ khi đã đăng nhập. 39](#_Toc107268927)

[**Hình 13** Màn hình đăng nhập. 39](#_Toc107268928)

[**Hình 14** Màn hình đăng ký tài khoản. 40](#_Toc107268929)

[**Hình 15** Màn hình tạo mới bài viết. 40](#_Toc107268930)

[**Hình 16** Màn hình bình luận, thích hoặc đánh giá bài viết. 41](#_Toc107268931)

[**Hình 17** Màn hình lọc nơi bạn muốn đến. 41](#_Toc107268932)

**Danh mục bảng**

[**Bảng 1** Đặc tả use case Đăng Ký. 22](#_Toc107433732)

[**Bảng 2** Đặc tả use case Đăng Nhập. 23](#_Toc107433733)

[**Bảng 3** Đặc tả use case tương tác với bài viết. 24](#_Toc107433734)

[**Bảng 4** Đặc tả use case Tạo Mới Bài Viết. 25](#_Toc107433735)

[**Bảng 5** Đặc tả use case quản lý của Admin. 27](#_Toc107433736)

[**Bảng 6** Những đặc điểm của ReactJS. 30](#_Toc107433737)

[**Bảng 7** Các thực thể. 48](#_Toc107433738)

[**Bảng 8** Tên các trường của bảng User. 50](#_Toc107433739)

[**Bảng 9** Tên các trường của bảng Post. 51](#_Toc107433740)

[**Bảng 10** Tên các trường của bảng PostRates. 51](#_Toc107433741)

[**Bảng 11** Tên các trường của bảng Comment. 52](#_Toc107433742)

[**Bảng 12** Danh sách thư viện được sử dụng. 53](#_Toc107433743)

[**Bảng 13** Danh sách các testcase mẫu. 59](#_Toc107433744)

# Danh mục các từ viết tắt

|  |  |
| --- | --- |
| **API** | Application Programming Interface.  Giao diện lập trình ứng dụng. |
| **EUD** | End-User Development.  Phát triển ứng dụng người dùng cuối. |
| **SQL** | Structured Query Language.  Ngôn ngữ truy vấn mang tính cấu trúc, được thiết kế để quản lý dữ liệu trong RDBMS. |
| **HTML** | HyperText Markup Language.  Ngôn ngữ đánh dấu siêu văn bản. |
| **URI** | Uniform Resource Identifier.  Một chuỗi các ký tự dùng để xác định một tài nguyên. |
| **JSON** | JavaScript Object Notation.  Một kiểu định dạng dữ liệu trong ngôn ngữ JavaScript. |
| **REST** | Representational State Transfer.  Một chuẩn thiết kế phần mềm, nó quy định cách mà  client và server sẽ tương tác với nhau. |
| **CNTT** | Công nghệ thông tin. |
| **v.v** | Vân vân. |
| **ĐATN** | Đồ án tốt nghiệp. |

# Danh mục thuật ngữ

|  |  |
| --- | --- |
| **Browser** | Trình duyệt |
| **Cache memory** | Bộ nhớ đệm |
| **E-commerce** | Thương mại điện tử |
| **Bloatware** | Ứng dụng nhà sản xuất tích hợp vào thiết bị |
| **Interpreter** | Trình thông dịch |
| **Compiler** | Trình biên dịch |
| **JavaScript** | Ngôn ngữ lập trình |
| **Backend** | Máy chủ |
| **Frontend** | Giao diện |
| **Web** | Trang mạng |
| **Render** | Quá trình hiển thị trang web bắt đầu từ lúc bạn nhận  được dữ liệu từ máy chủ |
| **Model** | Cấu trúc dữ liệu theo cách tin cậy |
| **Controller** | Bộ phận nhận lệnh, gửi lệnh đến cho Model để thực  hiện tương tác với dữ liệu |
| **Framework** | Bộ khung cung cấp các chức năng, giải pháp được cài đặt sẵn giúp tiết kiệm thời gian trong quá trình phát triển  ứng dụng. |

# Chương 1: Giới thiệu đề tài

* 1. **Đặt vấn đề**

Việt Nam đất nước được thiên nhiên ban tặng nhiều danh lam thắng cảnh đẹp. Đó những điểm đến du lịch hấp dẫn đối với khách trong nước và đặc biệt là khách du lịch là người nước ngoài. Tính trung bình năm 2015, tổng lượng khách du lịch nội địa lên đến 57 triệu lượt và 7,94 triệu lượt khách quốc tế (theo Tổng cục Du lịch Việt Nam).

Hiện nay những website ([dulichvietnam.com.vn](https://www.dulichvietnam.com.vn/), [travel.com.vn](https://travel.com.vn/), skydoor.net,…) và ứng dụng (Du Lịch Việt Nam S, 63 Tỉnh Thành Việt Nam,…) trên điện thoại di động đã giải quyết vấn đề về tra cứu thông tin du lịch nhưng vẫn chưa thực sự đầy đủ. Đi kèm với đó còn tồn tại các mặt hạn chế như sau:

* Nguồn thông tin còn mang tính quảng cáo nhiều khiến người dùng bị loãng thông tin.
* Nhiều những review về du lịch nhằm mục đích kinh doanh sẽ không chân thực dẫn đến nhiều hậu quả là thực tế không đúng với review.

Từ yêu cầu thực tế đó và cũng như phân tích từ những sản phẩm đi trước, em đã tìm hiểu cũng như đánh giá để đi đến giải pháp xây dựng Hệ thống hỗ trợ thông tin du lịch để khắc phục những hạn chế còn tồn động trên.

Trang web mà đề tài hướng đến sẽ cung cấp đầy đủ thông tin cho những ai muốn đi du lịch trong vùng miền lãnh thổ Việt Nam. Những chuyến đi và kinh nghiệm thực tế được người dùng chia sẻ cho những người khác, từ đó bạn có thể dễ dàng thiết kế lịch trình cho chuyến đi của mình qua những bài viết, blog được chia sẻ trên trang.

* 1. **Mục tiêu và phạm vi của đề tài**

Hiện nay các Website về thông tin du lịch khá nhiều, nhưng chỉ mang tính chất quảng cáo cho các dịch vụ du lịch. Việc tìm được thông tin chính xác về địa điểm, cảnh quan và những đánh giá khách quan từ phía người dùng đều gặp khó khăn. Hệ thống hỗ trợ thông tin du lịch được xây dựng trên cả nền web và ứng dụng di động để khắc phục những nhược điểm đó, ngoài ra còn cung cấp thêm các tiện ích khác sẽ nói rõ trong mô tả hệ thống.

Nhằm khắc phục những nhược điểm trên, website đưa ra hệ thống tìm kiếm những địa điểm du lịch mà người dùng hướng đến. Tại đó mọi người (có thể là admin hoặc người dùng) có thể chia sẻ kinh nghiệm mà họ đã trải qua hoặc đã có kinh nghiệm. Nó là trải nghiệm thực tế của người đi du lịch chứ không phải của KOL hay Blogger nào đó. Mọi người có thể đăng ký tài khoản sau đó comment những ý kiến góp ý của mình tại từng địa điểm mà họ đã đi qua. Đặc biệt nếu có những chia sẻ sâu, mang tính chuyên nghiệp thì người dùng có thể tạo các bài viết như một blog để review lại hành trình của mình cho mọi người một cách bài bản, bài viết đó sẽ được tạo bởi user và gửi đến Admin để quản trị viên xem xét và duyệt bài. Admin sẽ xem xét nội dung với các tiêu chí: lành mạnh, bổ ích, không quảng cáo để duyệt bài viết đưa đến cho người đọc tham khảo. Ở đó mọi người có thể tham khảo và chọn lọc những thông tin liên quan một cách dễ dàng và thực tế.

* 1. **Định hướng giải pháp**

Từ những vấn đề đã nêu ở mục 1.1 và 1.2, giải pháp đặt ra là phát triển hệ thống cho phép triển khai ứng dụng trên nền tảng website. Yêu cầu đặt ra là phát triển tách biệt phần backend và frontend, backend sử dụng với ngôn ngữ NodeJS sẽ cung cấp API để phát triển giao diện trên nền tảng web.

Phần backend của ứng dụng em phát triển là một dịch vụ web với kiến trúc RESTFULL, kiến trúc này rất phù hợp với việc phát triển các hệ thống lớn và việc mở rộng hệ thống sau này cũng dễ dàng hơn. Kiến trúc này cũng rất có ích cho hệ thống hiện tại, đáp ứng nhu cầu phát triển độc lập các thành phần. Ví dụ phát triển API em sử dụng NodeJS cụ thể là framwork NestJS.

Phần frontend em sử dụng kết hợp ReactJS. React là một thư viện viết bằng JavaScript, dùng để xây dựng giao diện người dùng nhiều tương tác phức tạp mà không phải load lại trang.

Phần dữ liệu em sử dụng cơ cơ dữ liệu MongoDB, nó cung cấp hiệu suất cao, tính khả dụng cao và khả năng mở rộng dễ dàng. Em dựa vào tỉ lệ đọc và ghi của dữ liệu để lựa chọn cách lưu trữ khác nhau sao cho hiệu suất truy vấn tốt nhất.

* 1. **Bố cục của đồ án**

Sau khi xác định được mục tiêu, phạm vi đề tài và định hướng giải pháp, phần còn lại của đồ án tốt nghiệp này được bố cục như sau.

Chương 2 đi vào phân tích xác định tổng quan chức năng và đặc tả cụ thể một số chức năng chính của ứng dụng hiển thị các bài viết review về du lịch và tìm kiếm những review nơi mình muốn đến. Cùng với đó là xác định những yêu cầu đi kèm như là bảo mật, tính xác thực của thông tin v.v. Chương 3 xác định những công nghệ cần sử dụng để có thể hoàn

thành những yêu cầu chức năng đặt ra ở chương 2. Phân tích những ưu điểm của công nghệ đó và lý do tại sao lại sử dụng những công nghệ đó cho việc phát triển ứng dụng.

Tiếp theo sau khi xác định được những chức năng cụ thể của ứng và những công nghệ sử dụng. Chương 4 trình bày về việc kết hợp những công nghệ đã trình bày ở chương 3 để tạo nên kiến trúc phần backend và fronend. Phần backend và frontend sẽ được phát triển độc lập, giao tiếp với nhau thông qua API. Từ kiến trúc đó đi sâu trình bày việc phát triển từng chức năng của ứng dụng, quá trình kiểm thử đi đến kết quả hoàn thiện. Chương 5 sẽ trình bày về những kiến trúc hệ thống nổi bật và kết quả, chức năng đặc biệt của ứng dụng.

Cuối cùng, Chương 6 tổng hợp lại quá trình phát triển ứng dụng, những kết quả đạt được, những điều chưa làm được, những bài học kinh nghiệm cần rút ra. So sánh kết quả ứng dụng hoàn thành với những ứng dụng tương tự đã có mặt trên mạng. Nêu ra định hướng phát triển của ứng dụng trong tương lai đi kèm với đó là những công việc, chức năng cần phát triển hoặc nâng cấp.

# Chương 2: Khảo sát và phân tích yêu cầu

* 1. **Khảo sát hiện trạng**

Hiện nay các Website về thông tin du lịch khá nhiều, nhưng chỉ mang tính chất quảng cáo cho các dịch vụ du lịch. Việc tìm được thông tin chính xác về địa điểm, cảnh quan và những đánh giá khách quan từ phía người dùng đều gặp khó khăn. Hệ thống hỗ trợ thông tin du lịch được xây dựng trên cả nền web và ứng dụng di động để khắc phục những nhược điểm đó, ngoài ra còn cung cấp thêm các tiện ích khác sẽ nói rõ trong mô tả hệ thống.

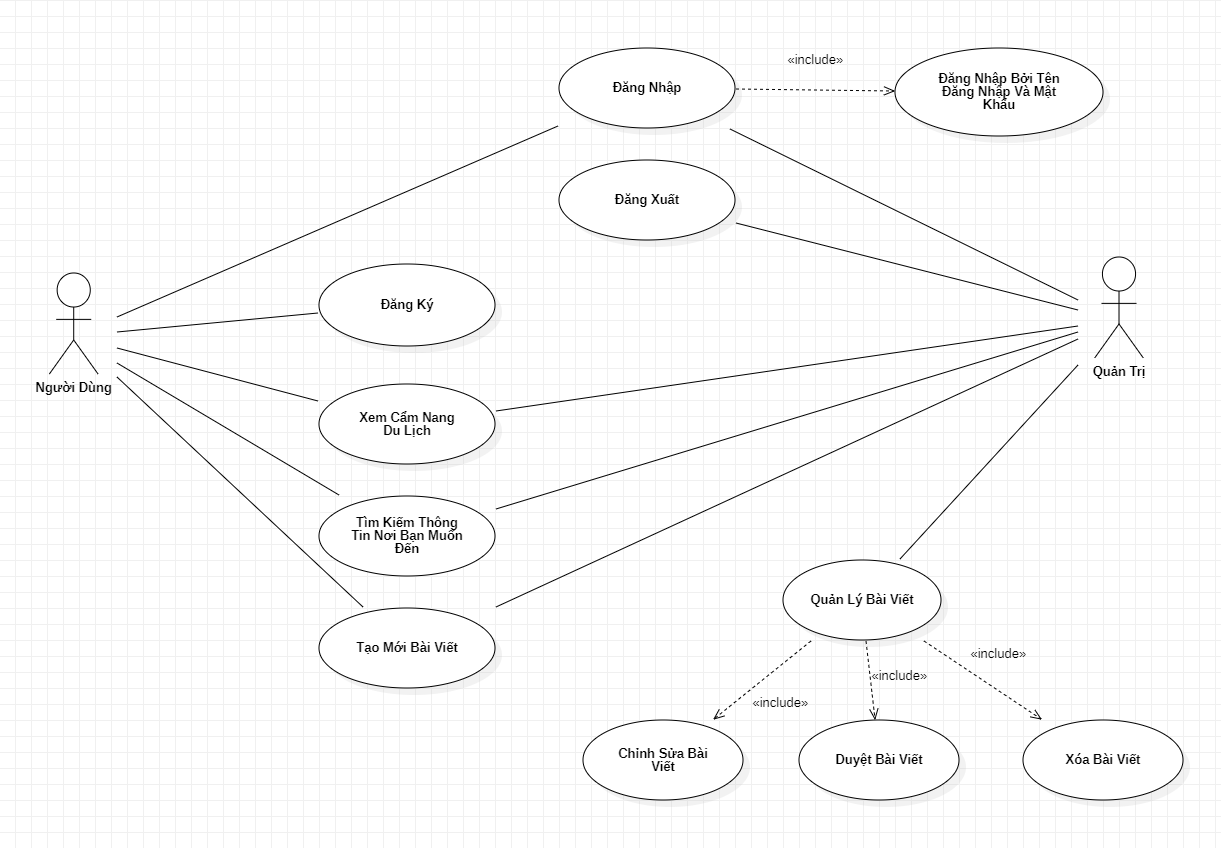
Người dùng có thể dễ dàng tham khảo cũng như chia sẻ các kinh nghiệm du lịch thực tế của mình cho mọi người biết đến với mục đích review chân thực và không quảng cáo. Ưu điểm này giúp cho mọi người cảm thấy tin tưởng và thích thú khi sử dụng website. Ở đây mỗi bài viết đều được chọn lọc và xét duyệt kỹ bởi quản trị viên nên đảm bảo những thông tin đến với người dùng đều chính xác và không mang tính chất quảng cáo.

Hơn nữa website với thiết kế tối giản rất thân thiện với người dùng kể cả với những người cao tuổi hoặc người không am hiểu chuyên sâu về công nghệ cũng có thể sử dụng.

* 1. **Tổng quan chức năng**

Phần này sẽ mô tả tóm tắt các chức năng của ứng dụng, các đặc tả chi tiết được trình bày trong phần 2.3.

* + 1. ***Biểu đồ use case tổng quan***



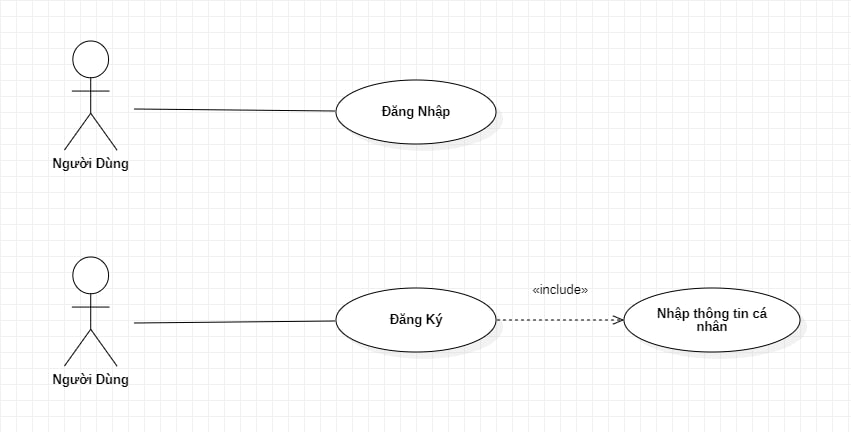
**Hình 1** Biểu đồ use case tổng quan.

**Hình 1** cho thấy trong ứng dụnng này có 2 tác nhân chính: người dùng và quản lý. Người dùng là khách đã đăng nhập, và có toàn bộ chức năng chính như: Xem cẩm nang du lịch,tìm kiếm nơi mình muốn đến, bình luận bài viết, tạo bài viết..vv . Quản lý là người dùng đã đăng nhập và xác thực có quyền quản lý, kế thừa toàn bộ chức năng của người dùng. Người quản lý có thể xem thống kê, quản lý số lượng bài viết, thống kê số lượng bài viết và số lượng người dùng của hệ thống.

* + 1. ***Biểu đồ use case phân rã***

Tiếp theo em sẽ phân rã một số use case chính trong ứng dụng.

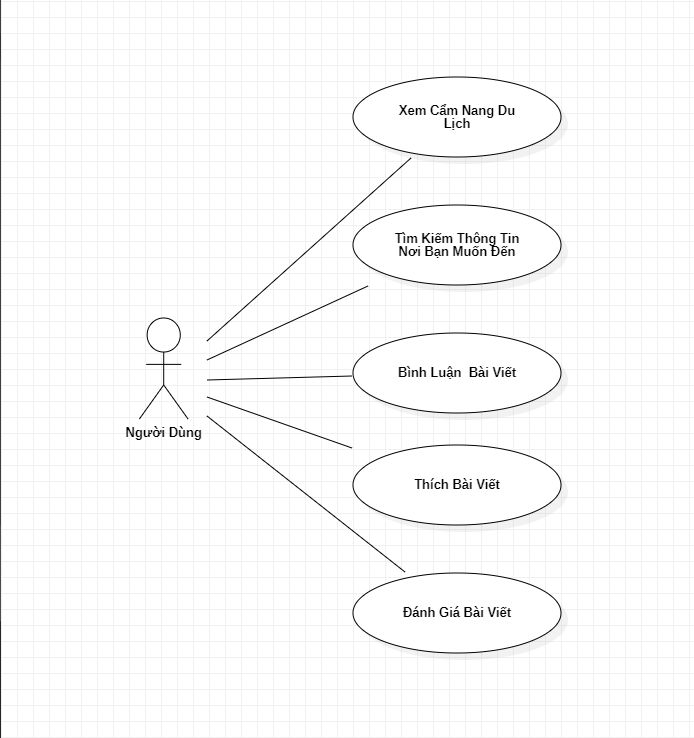
* + - 1. *Biểu đồ use case Đăng Nhập, Đăng Ký*



**Hình 2** Biểu đồ phân rã use case đăng nhập, đăng ký.

**Hình 2** giới thiệu use case phân rã của use case “Đăng Nhập, Đăng Ký”. Khi người dùng truy cập website nếu muốn các tương tác với bài viết thì cần đăng nhập tài khoản nếu đã có tài khoản hoặc đăng ký tài khoản nếu chưa có tài khoản. Người dùng đăng nhập hệ thống bằng tên và mật khẩu đã được đăng ký để có thể tương tác tốt hơn với website ngoài các chức năng cơ bản là xem bài biết và đọc thông tin.

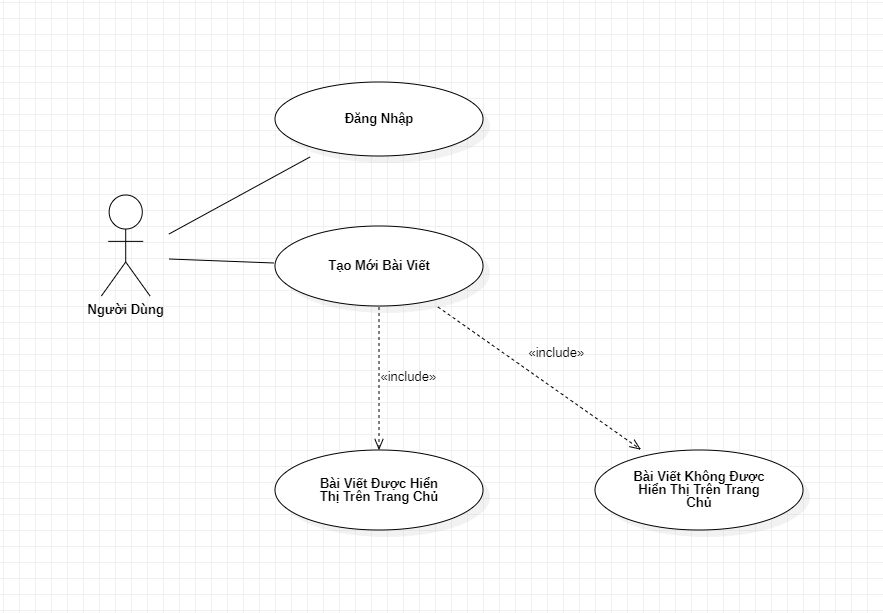
* + - 1. *Biểu đồ use case tương tác với bài viết*



**Hình 3** Biểu đồ phân rã use case tương tác với bài viết.

**Hình 3** giới thiệu các use case phân rã của use case “tương tác với bài viết”. Tương tác với bài viết ở đây là tương tác với tất cả các bài viết có trên ứng dụng chứ không riêng gì phim của bản thân người dùng tự tạoi. Người dùng có thể bình luận vào bài viết, thích bài viết, đánh giá về bài viết theo thang điểm từ 1 đến 5 sao, đặc biệt bình luận vềbài viết bằn chính tài khoản mà người dùng đã đăng ký trước đó.

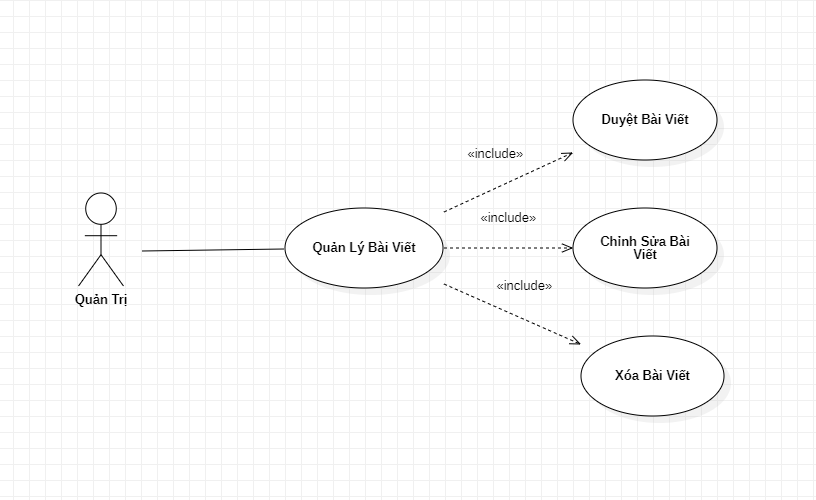
* + - 1. *Biểu đồ use case tạo mới bài viết*



**Hình 4** Biểu đồ phân rã use case tạo mới bài viết.

**Hình 4** giới thiệu về use case phân rã của use case tạo mới bài viết: Khi một người dùng đã đăng ký tài khoản và đăng nhập vào hệ thống bằng tài khoản đó, thì có thể thể thực hiện tạo mới bài viết để review về chuyển đi cũng như trải nghiệm của bản thân về những chuyến đi. Tiếp theo đó, bài viết sẽ được hiển thị trong Admin danh sách bài viết, quản trị viên sẽ xem xét mức độ cũng như tính đúng đắn và xác thực của bài viết để duyệt nó được hiển thị trên blog hay không. Nếu bài viết được quản trị viên phê duyệt thì sẽ được hiển thị trên trang chủ của website và ngược lại nếu bài viết không được phê duyệt thì sẽ không hiển thị trên trang chủ.

* + - 1. *Biểu đồ use case quản lý bài viết của Admin*



**Hình 5** Biểu đồ phân rã use case quản lý bài viết của Admin.

**Hình 5** giới thiệu các use case phân rã của use case “quản lý bài viết” của quản trị viên. Đối với những bài viết đã được người dùng đăng tải lên,bài viết sẽ hiển thị trong phần danh sách bài viết của Admin bao gồm Tên bài viết, Người đăng, Thời gian và Trạng thái của bài đăng. Từ đó quản trị viên có thể quản lý bằng các chỉnh sửa, phê duyệt hoặc xóa bài viết đó nếu nội dung bài viết không phù hợp với tiêu chí của website.

* 1. **Đặc tả chức năng**

Phần này em sẽ đưa ra đặc tả chi tiết của một số chức năng chính.

* + 1. ***Đặc tả use case đăng nhập, đăng ký***
       1. *Đặc tả use case đăng ký*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã use case** | **UC001** | | **Tên use case** | **Đăng Ký** |
| Tác nhân | Khách xem | | | |
| Tiền điều kiện | Khách xem chưa có tài khoản truy cập vào website | | | |
| Luồng sự kiện chính (thành công) | **STT** | **Thực hiện bởi** | | **Hành động** |
| 1 | Tác nhân | | Nhấn nút đăng ký |
| 2 | Hệ thống | | Chuyển đến màn đăng ký |
| 3 | Tác nhân | | Nhập thông tin |
| 4 | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 5 | Tác nhân | | Nhấn nút Đăng Ký |
| 6 | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 7 | Hệ thống | | Gọi use case Đăng Ký |
| 8 | Hệ thống | | Chờ đợi kết quả để hoàn tất |
| 9 | Hệ thống | | Thông báo đăng ký thành công |
| Luồng sự kiện rẽ nhánh | **STT** | **Thực hiện bởi** | | **Hành động** |
| 3a | Tác nhân | | Nhập thông tin không hợp lệ |
| 4a | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 5a | Tác nhân | | Nhấn nút Đăng Ký |
| 6a | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 7a | Hệ thống | | Hiển thị thông báo lỗi đối với các trường có thông tin không hợp lệ |
| Hậu điều kiện | Khách hàng trở thành thành viên của hệ thống | | | |

**Bảng 1** Đặc tả use case Đăng Ký.

* + - 1. *Đặc tả use case đăng nhập*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã use case** | **UC001** | | **Tên use case** | **Đăng Nhập** |
| Tác nhân | Người dùng | | | |
| Tiền điều kiện | Người dùng đã có tài khoản truy cập vào website | | | |
| Luồng sự kiện chính (thành công) | **STT** | **Thực hiện bởi** | | **Hành động** |
| 1 | Tác nhân | | Nhấn nút đăng nhập |
| 2 | Hệ thống | | Chuyển đến màn đăng nhập |
| 3 | Tác nhân | | Nhập thông tin đăng nhập |
| 4 | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 5 | Tác nhân | | Nhấn nút đăng nhập |
| 6 | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 7 | Hệ thống | | Gọi use case Đăng Nhập |
| 8 | Hệ thống | | Chờ đợi kết quả để hoàn tất |
| 9 | Hệ thống | | Chuyển đến màn hình trang chủ của website |
| Luồng sự kiện rẽ nhánh | **STT** | **Thực hiện bởi** | | **Hành động** |
| 3a | Tác nhân | | Nhập thông tin không hợp lệ |
| 4a | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 6a | Tác nhân | | Nhấn nút Đăng Nhập |
| 7a | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 8a | Hệ thống | | Hiển thị thông báo lỗi đối với các trường có thông tin không hợp lệ |
| Hậu điều kiện | Thành viên đã đăng nhập thành công và có thể sử dụng các chức năng mà hệ thống cung cấp. | | | |

**Bảng 2** Đặc tả use case Đăng Nhập.

* + 1. ***Đặc tả use case tương tác với bài viết***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã use case** | **UC001** | | **Tên use case** | **Tương tác với bài viết** |
| Tác nhân | Người dùng | | | |
| Tiền điều kiện | Người dùng đã đăng nhập vào website | | | |
| Luồng sự kiện chính (thành công) | **STT** | **Thực hiện bởi** | | **Hành động** |
| 1 | Tác nhân | | Truy cập vào bài viết |
| 2 | Hệ thống | | Chuyển đến màn hình chi tiết bài viết |
| 3 | Tác nhân | | Nhấn vào nút Thích bài viết |
| 4 | Tác nhân | | Nhấn vào đánh giá bài viết |
| 5 | Tác nhân | | Bình luận bài viết |
| 6 | Hệ thống | | Tiếp nhận thông tin người dùng vừa thao tác |
| 7 | Hệ thống | | Hiển thị người dùng đã thích bài viết |
| 8 | Hệ thống | | Hiển thị kết quả đánh giá bài viết của người dùng |
| 9 | Hệ thống | | Hiển thị bình luận của người dùng |
| Hậu điều kiện | Người dùng có thể tương tác với bài viết bằng các chức năng hiện có. | | | |

**Bảng 3** Đặc tả use case tương tác với bài viết.

* + 1. ***Đặc tả use case tạo mới bài viết***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã use case** | **UC001** | | **Tên use case** | **Tạo mới bài viết** |
| Tác nhân | Người dùng | | | |
| Tiền điều kiện | Người dùng đã đăng nhập vào website | | | |
| Luồng sự kiện chính (thành công) | **STT** | **Thực hiện bởi** | | **Hành động** |
| 1 | Tác nhân | | Nhấn nút tạo mới bài viết |
| 2 | Hệ thống | | Chuyển đến màn tạo mới bài viết |
| 3 | Tác nhân | | Nhập thông tin để tạo bài viết |
| 4 | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 5 | Tác nhân | | Nhấn nút Nộp |
| 6 | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 7 | Hệ thống | | Gọi use case Tạo Mới bài viết |
| 8 | Hệ thống | | Chờ đợi kết quả để hoàn tất |
| 9 | Hệ thống | | Chuyển đến màn hình trang chủ của website |
| Luồng sự kiện rẽ nhánh | **STT** | **Thực hiện bởi** | | **Hành động** |
| 3a | Tác nhân | | Nhập thông tin không hợp lệ |
| 4a | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 6a | Tác nhân | | Nhấn nút Nộp |
| 7a | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 8a | Hệ thống | | Hiển thị thông báo lỗi đối với các trường có thông tin không hợp lệ |
| Hậu điều kiện | Các thông tin bài viết được cập nhật vào cơ sở dữ liệu và hiển thị trong phần Dashboard của trang quản trị. | | | |

**Bảng 4** Đặc tả use case Tạo Mới Bài Viết.

* + 1. ***Đặc tả use case quản lý bài viết của Admin***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã use case** | **UC001** | | **Tên use case** | **Tương tác với bài viết** |
| Tác nhân | Quản trị viên | | | |
| Tiền điều kiện | Đăng nhập bằng tài khoản của quản trị viên | | | |
| Luồng sự kiện chính (thành công) | **STT** | **Thực hiện bởi** | | **Hành động** |
| 1 | Tác nhân | | Nhấn nút Dashboard |
| 2 | Hệ thống | | Chuyển đến màn hình Dashboard |
| 3 | Hệ thống | | Hiển thị danh sách bài viết đã đươc tạo bởi người dùng |
| 4 | Tác nhân | | Xóa bài viết |
| 5 | Hệ thống | | Bài viết bị xóa |
| 6 | Tác nhân | | Duyệt bài viết |
| 7 | Hệ thống | | Bài viết được hiển thị trên trang chủ của website |
| 8 | Tác nhân | | Chỉnh sửa thông tin bài viết |
| 9 | Hệ thống | | Hiển thị màn hình để chỉnh sửa thông tin bài viế |
| 10 | Tác nhân | | Nhập thông tin cần chỉnh sửa |
| 11 | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 12 | Tác nhân | | Nhấn nút Nộp |
| 13 | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 14 | Hệ thống | | Chờ đợi kết quả để hoàn tất |
| 15 | Hệ thống | | Chuyển đến màn hình trang chủ của website |
| Luồng sự kiện rẽ nhánh | **STT** | **Thực hiện bởi** | | **Hành động** |
| 8a | Tác nhân | | Nhập thông tin không hợp lệ |
| 9a | Hệ thống | | Hiển thị thông tin người dùng đã nhập |
| 10a | Tác nhân | | Nhấn nút Nộp |
| 11a | Hệ thống | | Kiểm tra thông tin người dùng vừa nhập |
| 12a | Hệ thống | | Hiển thị thông báo lỗi đối với các trường có thông tin không hợp lệ |
| Hậu điều kiện | Trạng thái bài viết sẽ được thay đổi và hiển thị theo cập nhật của quản trị viên. | | | |

**Bảng 5** Đặc tả use case quản lý của Admin.

* 1. **Yêu cầu phi chức năng (Bảo mật của API)**

Giao diện lập trình ứng dụng (API) là một phần mềm trung gian cho phép các ứng dụng của bạn giao tiếp với nhau.

Các API được xây dựng với Chuyển trạng thái biểu diễn (REST) ​​hoặc Giao thức truy cập đối tượng đơn giản (SOAP). REST nổi tiếng với các kỹ thuật đơn giản và nó có phong cách kiến ​​trúc đơn giản để xây dựng các dịch vụ web. Mặt khác, SOAP là một giao thức tin nhắn cho phép giao tiếp liền mạch giữa các phần tử của một ứng dụng.

Các API REST hoạt động với bảo mật lớp truyền tải và HTTP và cũng có thể sử dụng Ký hiệu đối tượng Javascript (JSON), trong khi SOAP hoạt động chủ yếu với Giao thức truyền siêu văn bản (HTTP).

API rất cần thiết cho các tương tác dựa trên web và do đó, đã trở thành mục tiêu của tội phạm mạng và tin tặc. Do đó, các phương pháp nhận dạng cơ bản như mật khẩu và tên người dùng đang được thay thế bằng mã thông báo bảo mật và xác thực đa yếu tố. Đây là cách bảo mật API hoạt động.

Chức năng bảo mật API chủ yếu với sự trợ giúp của ủy quyền và xác thực.

Xác thực là quy trình đầu tiên liên quan đến bảo mật API và nó xác minh rằng quy trình ứng dụng của bạn có danh tính an toàn cho phép bạn sử dụng API. Mặt khác, ủy quyền là bước tiếp theo xác định loại dữ liệu mà ứng dụng đã xác thực có quyền truy cập trong khi giao tiếp với API

Ngoài việc xác thực và ủy quyền an toàn, các API được phát triển với nhiều tính năng hơn để bảo vệ chúng và giảm nguy cơ bị tấn công từ nước ngoài. Một số tính năng như sau:

* Mã hóa thông báo bảo mật
* Mã hóa và Chữ ký
* Hạn ngạch và điều chỉnh
* Cổng API

Lợi ích của bảo mật API:

* API không phụ thuộc vào công nghệ
* Bảo mật API thoát khỏi lỗ hổng bảo mật
* Kết quả nhanh hơn
* Các lỗi có thể được phát hiện mà không cần thông báo của bạn

**Chương 3: Công nghệ sử dụng**

Chương 2 đã thực hiện khảo sát và đưa ra những yêu cầu về mặt chức năng và phi chức năng của ứng dụng. Xây dựng được biểu đồ use case tổng quan và phân rã của ứng dụng. Từ cái nhìn tổng quan đó chương 3 này sẽ tìm hiểu về giải pháp công nghệ thích hợp để có thể xây dựng hệ thống đáp ứng thực hiện được những chắc năng đã nêu ở chương 2.

1. **ReactJS**
2. ***Khái niệm về ReactJS***

React (còn được gọi là Reactjs hay React.js) là một Thư viện javascript được tạo ra bởi sự cộng tác giữa Facebook và Instagram. Nó cho phép những nhà phát triển web tạo ra giao diện người dung nhanh chóng. Phần Views của Reactjs thường được hiển thị bằng việc chủ yếu dung các component mà chứa các component cụ thể hoặc các thẻ HTML. Một trong những đặc trưng duy nhất của Reactjs là việc render dữ liệu không những có thể thực hiện ở tầng server mà còn ở tầng client.

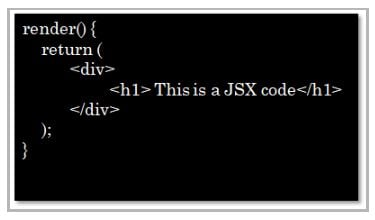
1. ***Đặc trưng của ReactJS***
2. *Những đặc điểm của React*

|  |  |
| --- | --- |
| **JSX** | JSX là một phần mở rộng cú pháp cho JavaScript. JSX được sử dụng với React để mô tả giao diện người dùng trông như thế nào. Bằng cách sử dụng JSX, chúng ta có thể viết các cấu trúc HTML trong cùng một tệp chứa code JavaScript. |
| **Thành phần (Components)** | Components là nền tảng của bất kỳ ứng dụng React nào và một ứng dụng đơn lẻ thường bao gồm nhiều components. Nó chia giao diện người dùng thành các phần độc lập, có thể tái sử dụng và có thể được xử lý riêng. |
| **DOM ảo (Virtual DOM)** | React giữ một bản “đại diện” nhưng nhẹ hơn của DOM “thực” trong bộ nhớ, gọi là DOM ảo (Virtual DOM). Khi trạng thái của một đối tượng (object) thay đổi, DOM ảo chỉ thay đổi đối tượng đó trong DOM thực, thay vì cập nhật tất cả các đối tượng. |
| **Truyền dữ liệu một chiều (One-way data-binding)** | Truyền dữ liệu một chiều (one-way data binding) của React giữ cho mọi thứ đều theo modular và nhanh chóng. Luồng dữ liệu một chiều có nghĩa là khi thiết kế một ứng dụng React, bạn thường lồng các components con bên trong các components mẹ. |
| **Hiệu suất cao** | React chỉ cập nhật những components đã thay đổi, thay vì cập nhật tất cả các components cùng một lúc. Điều này giúp tạo ra các ứng dụng web nhanh hơn đáng kể. |

**Bảng 6** Những đặc điểm của ReactJS.

1. *JSX*

JSX là một phần mở rộng cú pháp cho JavaScript. JSX được sử dụng với React để mô tả giao diện người dùng trông như thế nào. Bằng cách sử dụng JSX, chúng ta có thể viết các cấu trúc HTML trong cùng một tệp chứa code JavaScript



**Hình 6** Cấu trúc HTML sử dụng JSX.

Trình duyệt web không thể đọc JSX một cách trực tiếp. Điều này là do chúng được xây dựng để chỉ đọc các đối tượng JS thông thường và JSX không phải là một đối tượng JavaScript thông thường.

Để trình duyệt web có thể đọc được tệp JSX, tệp cần được chuyển đổi thành một đối tượng JavaScript thông thường. Để làm điều này, chúng tôi sử dụng Babel.

1. *Virtual DOM*

**DOM** là tên gọi tắt của **Document Object Model** (Mô hình Đối tượng Tài liệu), là một chuẩn được định nghĩa bởi W3C dùng để truy xuất và thao tác trên code HTML hay XML bằng các ngôn ngữ lập trình thông dịch (scripting language) như Javascript.

**DOM** giúp thao tác với dữ liệu theo mô hình hướng đối tượng do các phần tử trong **DOM** có cấu trúc được định nghĩa thành các đối tượng, phương thức, thuộc tính để có thể truy xuất dễ dàng. Chúng được coi như các node và được biểu diễn dưới dạng **DOM Tree.**

**Virtual DOM** không được tạo ra bởi React tuy nhiên nó được React sử dụng và cung cấp miễn phí.



**Hình 7** Cấu trúc HTML DOM.

1. *Component*

React được xây dựng xung quanh các thành phần (component), chứ không dùng template như các framework khác. Trong React, chúng ta xây dựng trang web sử dụng những 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. Hai thành phần đặc biệt quan trọng khi dùng component là props và state.

* Props giúp các component tương tác với nhau, component nhận đầu vào là props, 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 chỉ component có state đó đồng thời render lại để cập nhật UI.

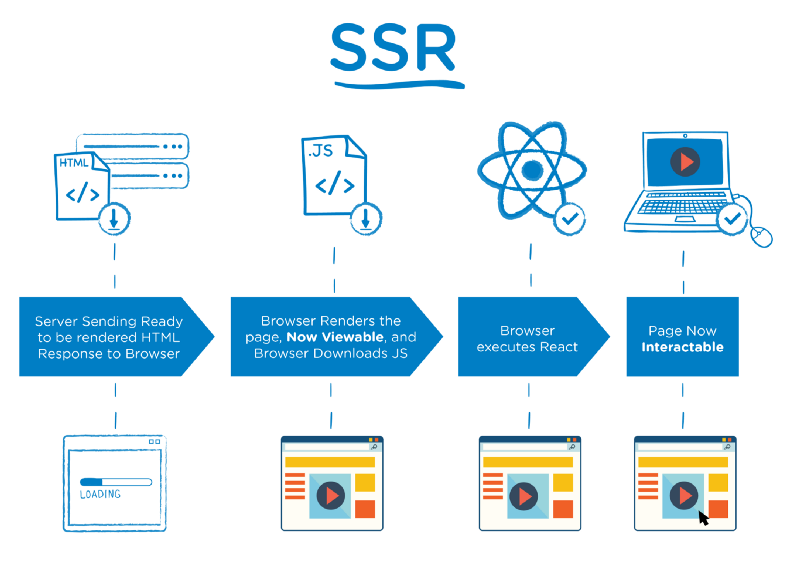
1. *Server side rendering*

Nội dung HTML được xử lý trên server và trả lại cho browser, lúc này nội dung HTML đã hoàn chỉnh và được hiển thị ngay khi nó được load về máy, tuy nhiên lúc này trang web mới chỉ xem được nội dung, người dùng chưa thể tương tác được. Ngay trong lúc này, trình duyệt vẫn âm thầm tải tiếp JS và thực thi nó ngay khi hoàn tất, một khi hoàn tất quá trình này người dùng mới có thể tương tác được.

Nó sẽ không là vấn đề nếu bạn truy cập 1 trang mới, mà trang này chỉ có 1 số nội dụng thay đổi so với trang hiện tại, lúc này browser sẽ gửi request lên server, server nhận được yêu cầu và re-render lại toàn bộ trang mới.

Ưu điểm của Server side rendering:

* Initial load nhanh, dễ otpimize, vì toàn bộ dữ liệu đã được xử lý ở server, client chỉ việc hiển thị.
* Các web framework từ xưa đến nay đều hỗ trợ cơ chế này.
* Dễ hiểu và dễ code hơn. Chỉ cần code 1 project web là được, không cần phải tách ra frontend và backend.
* SEO tốt vì khi bot của Google, Bing vào web sẽ thấy toàn bộ dữ liệu dưới dạng HTML.
* Chạy được trên phần lớn mọi trình duyệt, kể cả disable JavaScript vẫn chạy tốt.



**Hình 8** Quá trình thực hiện server side rendering.

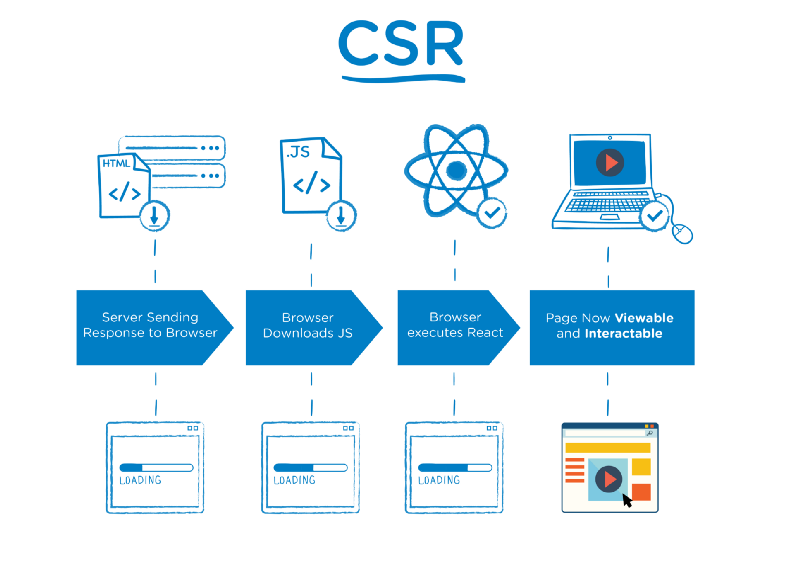
1. *Client side rendering*

Client side rendering tức là việc render HTML, CSS sẽ được thực hiện ở client tức là chạy JavaScript ở trình duyệt. Khi request lên server, server sẽ trả về file HTML và file nén JavaScript riêng biệt, sau khi client tải 2 file về hoàn tất, browser sẽ thực hiện render các component và thực hiện lấy dữ liệu rồi đổ ra màn hình. Những logic đơn giản sẽ nằm ở client side, logic để routing (chuyển trang), render (hiển thị) dữ liệu đều năm ở client side còn những logic phức tạp vẫn nằm ở server side. Client side rendering giải quyết được một số vấn đề của server side rendering như:

* Page chỉ cần load một lần duy nhất. Khi user chuyển trang hoặc thêm dữ liệu, JavaScript sẽ lấy và gửi dữ liệu từ server qua một phương thức bất đồng bộ. Người dùng có thể thấy dữ liệu mới mà không cần chuyển trang.
* Chuyển logic sang client nên giảm tải được một phần cho server.
* Giảm được băng thông do chỉ cần lấy dữ liệu dạng JSON và dữ liệu cần thiết, thay vì phải lấy toàn bộ trang.
* Với các ứng dụng cần tương tác nhiều sẽ hoạt động mượt mà hơn vì code chạy trên browser, không cần load đi loại lại nhiều lần.

Nhược điểm của CRS:

* Page load lần đầu sẽ chậm hơn nếu không optimize chuẩn. Lý do là trình duyệt phải tải toàn bộ JavaScript về (sẽ khá nặng), phân tích và chạy JS , gọi API để lấy dữ liệu từ server (chậm), sau đó render dữ liệu
* Đòi hỏi project phải chia làm 2 phần riêng là back-end (REST api) và frontend nên khó code hơn
* Không chạy được nếu JavaScript bị disable, hoặc ở các trình duyệt cũ không nhận JavaScript ES6 (Có thể dùng transpiler và polyfill nhưng sẽ làm tăng kích cỡ file js)
* SEO không tốt bằng Server Side Rendering (Do bot crawl không đọc được dữ liệu). Để giải quyết, ta phải kết hợp thêm SSR (Bot mới của Google đọc được client-side rendering rồi).
* Nếu client sử dụng mobile, device yếu thì khi load sẽ bị chậm



**Hình 9** Quá trình thực hiện client side rendering.

1. *NextJS*

NextJS là một framework Node.JS cho phép xây dựng ứng dụng phía server. Nest mở rộng các framework Node.js như Express hay Fastify để bổ sung thêm nhiều module hay thư viện hỗ trợ việc xử lý tác vụ. Đây là một framework mã nguồn mở, sử dụng TypeScript và rất linh hoạt để xây dựng các hệ thống backend.

1. **Redux**
2. ***Giới thiệu về Redux***

Redux [7] là một thư viện giúp quản lý trạng thái của ứng dụng. Nó được thiết kế dựa trên Flux [8], nhưng giảm bớt những đau khổ thường gặp phải khi viết một ứng dụng Flux. Flux là một kiến trúc mà Facebook sử dụng trong khi làm việc với React, xây dựng ý tưởng về luồng dữ liệu một chiều. Redux thường là bộ đôi kết hợp hoàn hảo với React.

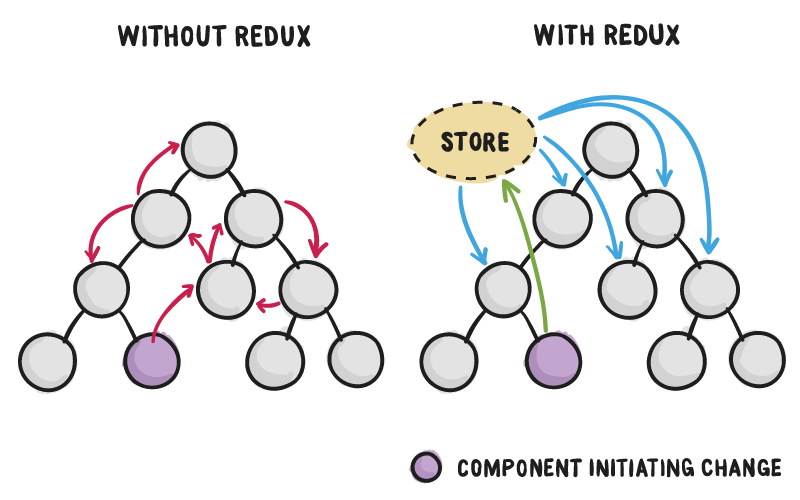
1. ***Redux và ReactJS***

Như đã trình bày ở trên về React Virtual DOM, Redux sẽ giúp cấu hình để quản lý những Virual DOM này một cách tổng quát. Hình 9 minh họa cho quá trình thay đổi dữ liệu của một sự kiện. Với việc không sử dụng redux, các component giao tiếp với nhau thông qua cách truyền props, khi một component mức cao muốn truyền một giá trị cho component ở mức thấp sẽ phải truyền tuần tự props qua các component ở mức trung gian, và ngược lại component mức thấp muốn thực hiện một event cũng phải truyền tuần tự ngược qua các component trung gian. Dễ thấy việc này khiến cho việc quản lý state và props trở lên phức tạp hơn khi dự án được mở rộng. Redux giúp ứng dụng xây dựng một store thay vì lưu trữ toàn bộ data tại một root component thì sẽ lưu nó tại store của redux, và nếu có thay đổi, đơn giản là ứng dụng sẽ thông qua redux và thay đổi dữ liệu trong strore. Store ở đây hiểu đơn giản giống như một biến toàn cục lưu trạng thái, dữ liệu của ứng dụng.

1. ***Hoạt động của Redux***

Redux gồm ba thành phần chính và nguyên tắc khi sử dụng là:

* Store (Object) là nguồn dữ liệu chân lý duy nhất. Chỉ được phép đọc state trong store, các duy nhất thay đổi state là phát sinh một action để reducer thay đổi state.
* Action (Function) là những mô tả cho từng event.
* Reducer (Function) là 1 hàm nhận đầu vào các initial state (trạng thái ban đầu của ứng dụng) và các mô tả về event và chỉ dựa trên đó để trả về state tiếp theo.



**Hình 10** Quản lý dữ liệu có redux và không có redux [9].

1. **MongoDB**
2. ***Giới thiệu về MongoDB***

NoSQL là 1 dạng cơ sở dữ liệu mã nguồn mở và được viết tắt bởi: None-Relational SQL hay có nơi thường gọi là Not-Only SQL. MongoDB [10] là một hệ quản trị cơ sở dữ liệu mã nguồn mở thuộc NoSQL. Nó được thiết kế theo kiểu hướng đối tượng và lưu trữ dữ liệu theo hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh. Các bảng trong mongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ trên bảng không cần tuân theo một cấu 23 trúc nhất định nào cả, điều này rất thích hợp để làm việc với dữ liệu lớn. Bảng 5 mô tả những thuật ngữ được dùng trong mongoDB:

Các thuật ngữ hay được sử dụng 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 mấy 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.

Chú thích: SQL: là viết tắt của Structured Query Language, là ngôn ngữ truy vấn mang tính cấu trúc. Nó được thiết kế để quản lý dữ liệu trong một hệ thống quản lý cơ sở dữ liệu quan hệ.

Kiểu dữ liệu đặc biệt trong mongoDB:

* **Object ID:** Kiểu dữ liệu này được sử dụng để lưu giữ ID của Document.
* **Min/Max keys:** Kiểu dữ liệu này được sử dụng để so sánh một giá trị với các phần tử BSON thấp nhất và cao nhất.
* **Array:** Kiểu dữ liệu này được sử dụng để lưu giữ các mảng hoặc danh sách hoặc nhiều giá trị vào trong một key trong field

1. ***Ưu nhược điểm của MongoDB***

**Ưu điểm:**

* Document oriented
* Hiệu suất cao
* Tính sẵn sàng cao – Nhân rộng
* Khả năng mở rộng cao – Sharding
* Năng động – Không có lược đồ cứng nhắc
* Linh hoạt – thêm / xóa trường có ít hoặc không ảnh hưởng đến ứng dụng
* Dữ liệu không đồng nhất
* Không joins
* Phân phối được
* Biểu diễn dữ liệu trong JSON hoặc BSON
* Hỗ trợ không gian địa lý (Geospatial)
* Tích hợp dễ dàng với BigData Hadoop
* Ngôn ngữ truy vấn dựa trên tài liệu mạnh mẽ như SQL
* Các bản phân phối cloud như AWS, Microsoft, RedHat, dotCloud và SoftLayer, v.v . . . Trên thực tế, MongoDB được xây dựng cho cloud. Kiến trúc mở rộng quy mô tự nhiên của nó, được kích hoạt bởi sharding, liên kết tốt với quy mô và sự nhanh nhẹn có được nhờ điện toán đám mây

**Nhược điểm:**

* Một nhược điểm của NoSQL là hầu hết các giải pháp đều không tuân thủ ACID mạnh mẽ (Atomic, Consistency, Isolation, Durability) như các hệ thống RDBMS được thiết lập tốt hơn.
* Giao dịch phức tạp
* Không có chức năng hoặc thủ tục lưu trữ tồn tại nơi bạn có thể liên kết logic

1. **NodeJS**
2. ***Giới thiệu về NodeJS***

Nodejs là một nền tảng (Platform) phát triển độc lập được xây dựng trên V8 JavaScript Engine – trình thông dịch thực thi mã JavaScript giúp chúng ta có thể xây dựng được các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp một cách nhanh chóng và dễ dàng mở rộng.

NodeJS có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ Window cho tới Linux, OS X nên đó cũng là một lợi thế. 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.

Node.js được xây dựng và phát triển từ năm 2009, bảo trợ bởi công ty Joyent, trụ sở tại California, Hoa Kỳ.

1. ***NodeJS hoạt động như thế nào?***

Ý tưởng chính của Node js là sử dụng non-blocking, hướng sự vào ra dữ liệu thông qua các tác vụ thời gian thực một cách nhanh chóng. Bởi vì, Node js có khả năng mở rộng nhanh chóng, khả năng xử lý một số lượng lớn các kết nối đồng thời bằng thông lượng cao.

Nếu như các ứng dụng web truyền thống, các request tạo ra một luồng xử lý yêu cầu mới và chiếm RAM của hệ thống thì việc tài nguyên của hệ thống sẽ được sử dụng không hiệu quả. Chính vì lẽ đó giải pháp mà Node js đưa ra là sử dụng luồng đơn (SingleThreaded), kết hợp với non-blocking I/O để thực thi các request, cho phép hỗ trợ hàng chục ngàn kết nối đồng thời.

1. ***MongooseJS***

MongooseJS [14] là một thư viện mô hình hóa đối tượng (Object Data Model - ODM) cho MongoDB và NodeJS. MongooseJS là cho phép định nghĩa các schema với dữ liệu được định kiểu rõ ràng. Khi một schema được định nghĩa, Mongoose cho phép tạo một Model dựa trên một schema cụ thể. Model của MongooseJS sau đó được ánh xạ tới một MongoDB document thông qua định nghĩa schema của Model.

Sau khi xác định các schema và model, MongooseJS chứa nhiều hàm khác nhau cho phép xác thực tính hợp lệ, lưu, xóa và truy vấn dữ liệu của bạn bằng các hàm MongoDB phổ biến.

1. ***Tailwind CSS***
   * 1. ***Tailwind CSS là gì ?***

Thì nó là một utility-first CSS framework, nó cũng giống như Bootstrap, nó có những class built-in mà chúng ta có thể dùng. Tailwind CSS có nhiều các class bao gồm các thuộc tính CSS khác nhau và quan trọng, chúng ta có thể dễ dàng mở rộng tạo mới ra những class bằng chính những class của nó.

* + 1. ***Tại sao chúng ra nên dùng Tailwind?***

Nói chung là nó cũng na ná Boostrap thôi nhưng một điều tiện lợi khi dùng framework này là chúng ta có nhiều class mới hơn tiện lợi hơn Boostrap. Các bạn có thể tham khảo tại đây trong Tailwind. Và hơn nữa, việc có nhiều thêm những class nhưng với quy tắc đặt tên cực kỳ thân thiện với người dùng, người dùng cũng có thể nhìn vào class đó và có thể biết được class này nó đang style cái gì. Chúng ta cũng phải nói đến khả năng tùy biến và mở rộng cao, đem đến cho ta sự linh hoạt không giới hạn.

1. **Bảo mật API với Json web token – JWT**
2. ***JSON Web Token là gì?***

JSON Web Token (JWT) la mọt chuân mơ (RFC 7519) đê truyên thong tin an toan giưa cac ben nhu mọt đôi tuơng JSON. Thong tin nay đuơc xac minh va đang tin cạy bơi chư ky sô. JWT đuơc gán với một khoá bí mật (sử dụng thuạt toan HMAC) hoạc mọt cạp khoa cong khai/ khoa rieng sư dung RSA hoặc ECDSA. Các thông tin trong chuỗi JWT được định dạng bằng Json.

1. ***Khi nào nên sử dụng JSON Web Token?***

Xác thực (Authentication): Tinh huông thuơng gạp nhât, khi người dùng đang nhạp, môi yeu câu tiêp theo đêu kem theo chuôi token JWT, cho phep nguơi dung co thê truy cạp đuơng dân, dich vu va tai nguyen đuơc phep ưng vơi token đo. Đang nhạp mọt lân cung la mọt chưc nang co sư dung JWT mọt cach rọng rai, bơi vi chuôi JWT co kich thuơc đu nho đê đinh kem trong yeu câu va sư dung ơ nhiêu hẹ thôngkhac nhau.

Trao đổi thông tin(Information Exchange): JSON Web Token cũng là một cách hữu hiệu và bảo mật để trao đổi thông tin giữa nhiều ứng dụng bởi JWT phải được ký bằng cặp khóa công khai và khóa riêng.Ngoài ra, ký tự cũng được tính toán dựa trên nội dung của 23 tiêu đề và tải nội dung, nhờ đó, bạn có thể xác định nội dung là bản nguyên, chưa được chỉnh sửa hoặc có thể đóng gói. Tuy nhiên, một lưu ý hết sức quan trọng là do đơn giản cấu trúc của JWT, JWT có thể dễ dàng bị giải mã, do đó, không nên sử dụng JWT để truyền thông tin.

Ví dụ về JWT token:

“eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjEzODY4OTkxMzEsImlzcyI6ImppcmE6MTU0ODk1OTUiLCJxc2giOiI4MDYzZmY0Y2ExZTQxZGY3YmM5MGM4YWI2ZDBmNjIwN2Q0OTFjZjZkYWQ3YzY2ZWE3OTdiNDYxNGI3MTkyMmU5IiwiaWF0IjoxMzg2ODk4OTUxfQ.uKqU9dTB6gKwG6jQCuXYAiMNdfNRw98Hw\_IWuA5MaMo”

1. ***Cấu trúc của Json Web Token***
2. *Header*

Phần header dùng để khai báo kiểu chữ ký và thuật toán mã hóa sẽ dùng cho cái token của API.

Ví dụ:

* “typ” (type) chỉ ra rằng đối tượng là một JWT
* “alg” (algorithm) xác định thuật toán mã hóa cho chuỗi là HS256

1. *Payload*

Payload chứa các claims. Claims là một các biểu thức về một thực thể (chẳng hạn user) và một số metadata phụ trợ. Có 3 loại claims thường gặp trong Payload: reserved, public và private claims.

Reserved claims: Đây là một số metadata được định nghĩa trước, trong đó một số metadata là bắt buộc, số còn lại nên tuân theo để JWT hợp lệ và đầy đủ thông tin: iss (issuer), iat (issued-at time) exp (expiration time), sub (subject), aud (audience), jti (Unique Identifier cho JWT, Can be used to prevent the JWT from being replayed. This is helpful for a one time use token.) ... Ví dụ:

{

"iss": "jira:1314039",

"iat": 1300819370,

"exp": 1300819380,

"qsh": "8063ff4ca1e41df7bc90c8ab6d0f6207d491cf6dad7c66ea797b4614b71922e9",

"sub": "batman",

"context": {

"user": {

"userKey": "batman",

"username": "bwayne",

"displayName": "Bruce Wayne"

}

}

}

{

"iss": "scotch.io",

"exp": 1300819380,

"name": "Chris Sevilleja",

"admin": true

}

Public Claims - Claims được cộng đồng công nhận và sử dụng rộng rãi.

Private Claims - Claims tự định nghĩa (không được trùng với Reserved Claims và Public Claims), được tạo ra để chia sẻ thông tin giữa 2 parties đã thỏa thuận và thống nhất trước đó.

1. *Signature*

Phần Signature sẽ được tạo ra bằng cách mã hóa phần header, payload kèm theo một chuỗi khoá bí mật (secret key).

Ví dụ nếu sử dụng thuật toán HMAC SHA256 để mã hoá header và payload, chữ ký sẽ được tạo theo cách sau:

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

Do bản thân Signature đã bao gồm cả header và payload nên Signature có thể dùng để kiểm tra tính toàn vẹn của dữ liệu khi truyền tải.

# Chương 4: Phát triển và triển khai ứng dụng

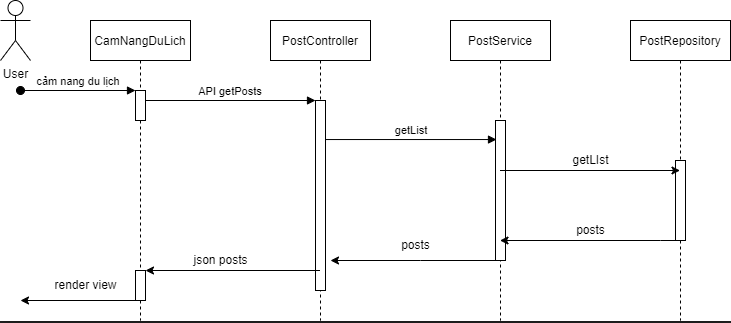
Chương 3 đã xác định được những công nghệ cần sử dụng để có thể đáp ứng được nhu cầu chức năng đặt ra ở chương 2. Chương 4 này sẽ đi sâu vào tổ chức kiến trúc hệ thống bằng cách kết hợp, tận dụng những ưu điểm của những công nghệ đã nêu ra ở chương 3. Từ kiến trúc hệ thống tiếp tục đi sâu vào thiết kế chi tiết từng thành phần, các luồng hoạt động chính. Xác định được những thành phần cần thiết từ đó thiết kế cơ sở dữ liệu và cuối cùng là quá trình triển khai hệ thống hoàn chỉnh và kết quả đạt được.

1. **Thiết kế kiến trúc**

Website được phát triển bao gồm 2 phần Backend và Fontend. Phần backend được phát triển dựa theo kiến trúc RESTful Web Service để trao đổi với phần frontend. Phần frontend em sử dụng kiến trúc Flux, việc hiển thị sẽ được chia ra thành các thành phần (component-based).

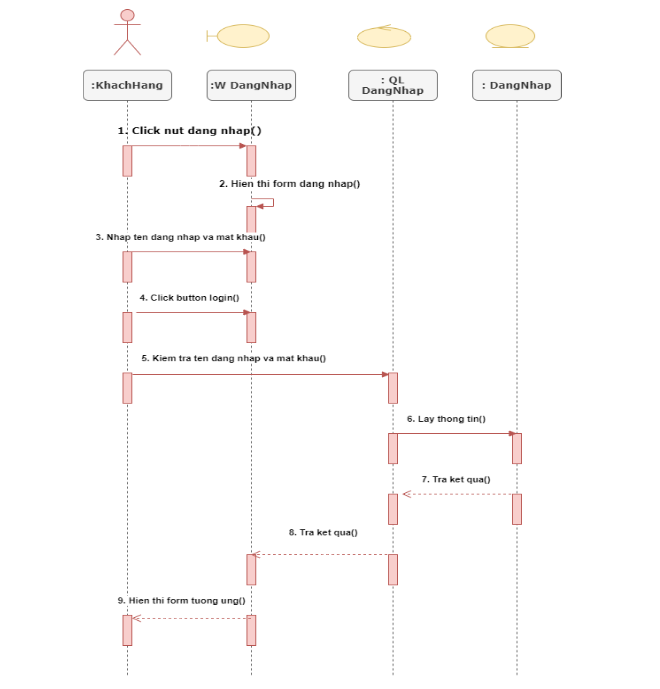
1. ***Biểu đồ tuần tự toàn hệ thống***

Biểu đồ tuần tự của hệ thống website:



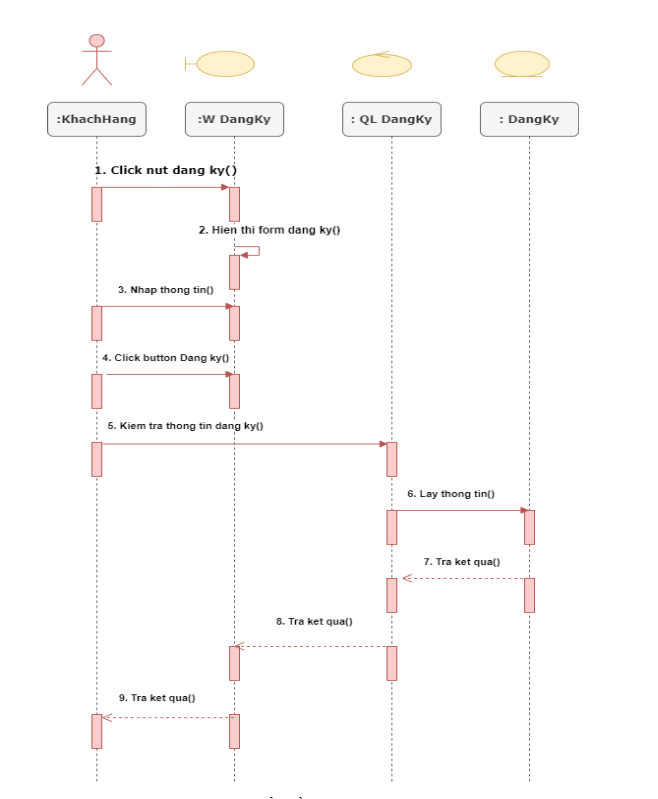
**Hình 11** Biểu đồ tuần tự của hệ thống.

1. ***Biểu đồ tuần tự UC Đăng nhập***



**Hình 12** Biểu đồ tuần tự UC Đăng nhập.

1. ***Biểu đồ tuần tự UC Đăng ký***



**Hình 13** Biểu đồ tuần tự UC Đăng ký.

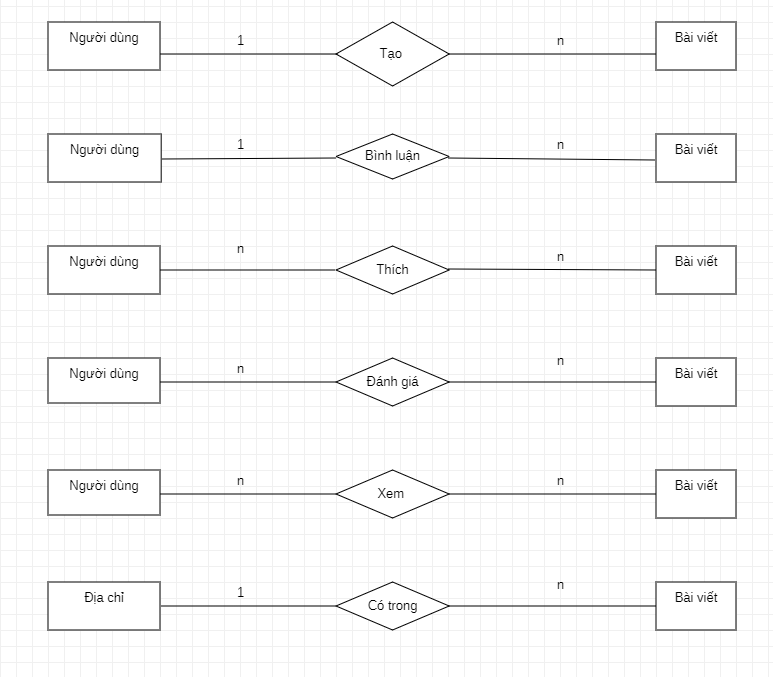
1. **Mô hình hóa dữ liệu**
2. ***Xác định thực thể***

Trong quá trình làm đồ án em đã phân tích hệ thống và tham khảo các trang web khác em sẽ đưa ra các thực thể trong bảng sau:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thực thể** | **Nội dung lưu trữ** |
| 1 | Người dùng | Thông tin người dùng gồm: tên đăng nhập và mật khẩu |
| 2 | Bài viết | Thông tin bài viết gồm các thuộc tính: tiêu đề, nội dung, hình ảnh, địa điểm |
| 3 | Địa chỉ | Địa chỉ có trong thông tin bài viết |

**Bảng 7** Các thực thể.

1. ***Xác định liên kết giữa các thực thể***

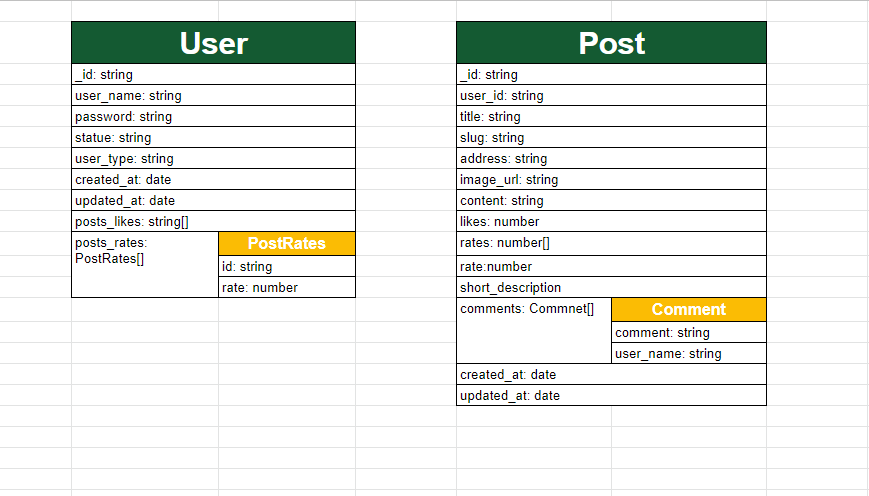


**Hình 14** Mối quan hệ giữa các thực thể.

1. ***Thiết kế cơ sở dữ liệu MongoDB***

Sau khi xác định được các thực thể và mối quan hệ giữa chúng, em tiến hành tổ chức và lưu trữ dữ liệu trong MongoDB. Do liên quan đến việc thu thập dữ liệu phục vụ bài toán gợi ý phim cho người dùng nên một số dữ liệu sẽ được lưu trữ một cách tách biệt. Dựa vào đặc điểm của dữ liệu để xác định lưu trữ theo một trong ba cách: (i) cách 1 là nhúng toàn bộ dữ liệu liên quan trong một document trong collection, (ii) cách 2 là tạo document là một mảng chứa ObjectId của collection con, (iii) cách 3 là lưu trữ mỗi document là một cặp các ObjectId.

Dưới đây là các bảng của database:



**Hình 15** Thiết kế DB cho ứng dụng.

1. Bảng User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Độ dài** | **Mô tả** |
| 1 | id | string | 20 | id người dùng |
| 2 | user\_name | string | 10 | tên người dùng |
| 3 | password | string | 32 | mật khẩu |
| 4 | statue | string | 20 | trạng thái của tài khoản |
| 5 | user\_type | string | 20 | loại người dùng |
| 6 | created\_at | date |  | ngày tạo |
| 7 | updated\_at | date |  | ngày cập nhật |
| 8 | posts\_likes | string[] |  | trạng thái thích/ không thích bài viết |

**Bảng 8** Tên các trường của bảng User.

1. Bảng Post

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Độ dài** | **Mô tả** |
| 1 | id | string | 20 | id của bài viết |
| 2 | user\_id | string | 20 | id người dùng |
| 3 | title | string | 50 | tiêu đề của bài viết |
| 4 | slug | string | 50 | địa chỉ truy cập bài viết |
| 5 | address | string | 150 | địa chỉ có thông tin trong bài viết |
| 6 | image\_url | string | 10MB | ảnh minh họa |
| 7 | likes | number |  | số lượng lượt thích |
| 8 | rates | number |  | số lượng đánh giá |
| 9 | short description | sting | 255 | nội dung ngắn gọn của bài viết |
| 10 | created\_at | date |  | ngày tạo bài viết |
| 11 | updated\_at | date |  | ngày cập nhật bà viết |

**Bảng 9** Tên các trường của bảng Post.

1. Bảng PostRates

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Độ dài** | **Mô tả** |
| 1 | id | string | 20 | id PostRates |
| 2 | rates | number |  | số lượng đánh giá |

**Bảng 10** Tên các trường của bảng PostRates.

1. Bảng comment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Độ dài** | **Mô tả** |
| 1 | comment | string | 1000 | bình luận của bài viết |
| 2 | user\_name | string | 10 | id người dùng |

**Bảng 11** Tên các trường của bảng Comment.

1. **Xây dựng ứng dụng**
2. ***Thư viện và công dụng sử dụng***

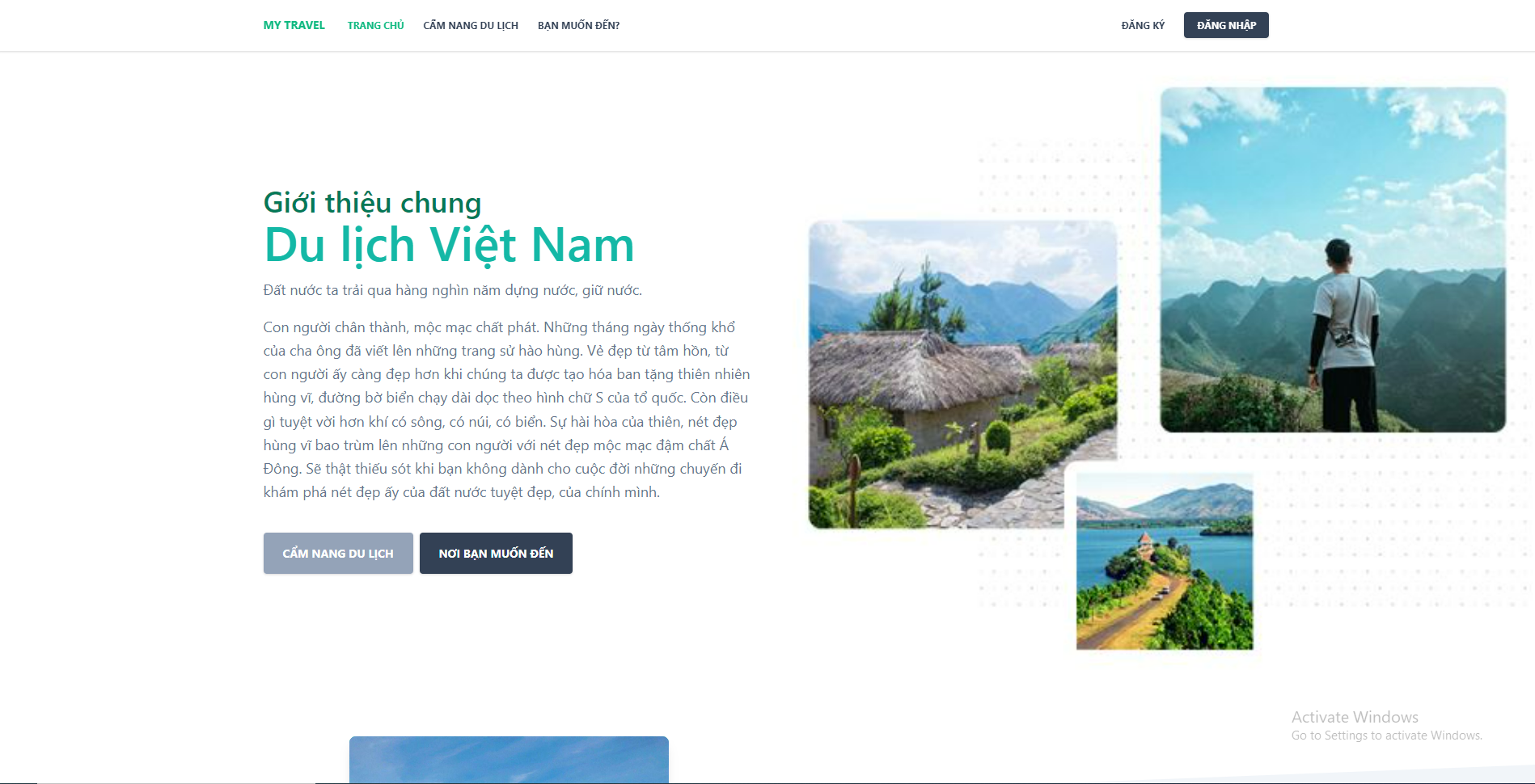
|  |  |  |
| --- | --- | --- |
| **Mục đích** | **Công cụ** | **URL** |
| IDE lập trình | Visual Studio Code 64 bit | https://code.visualstudio.com/ |
| Môi trường lập trình backend | NodeJS 10.15.3 | https://nodejs.org |
| Thư viện giúp mô hình hóa dữ liệu | MongooseJS 5.5.11 | https://mongoosejs.com/ |
| Xây dựng lưu trữ dữ liệu | MongoDB 4.0 | https://www.mongodb.com/ |
| Thư viện lập trình frontend | ReactJS 16.8.4 | https://reactjs.org/ |
| Thư viện giúp xây dựng kiến trúc Redux | Redux 4.0.1 | https://redux.js.org/ |
| Công cụ quản lý các thư viện lập trình NodeJS | Npm 6.9.0 | https://www.npmjs.com/ |

**Bảng 12** Danh sách thư viện được sử dụng.

1. ***Kết quả đạt được***

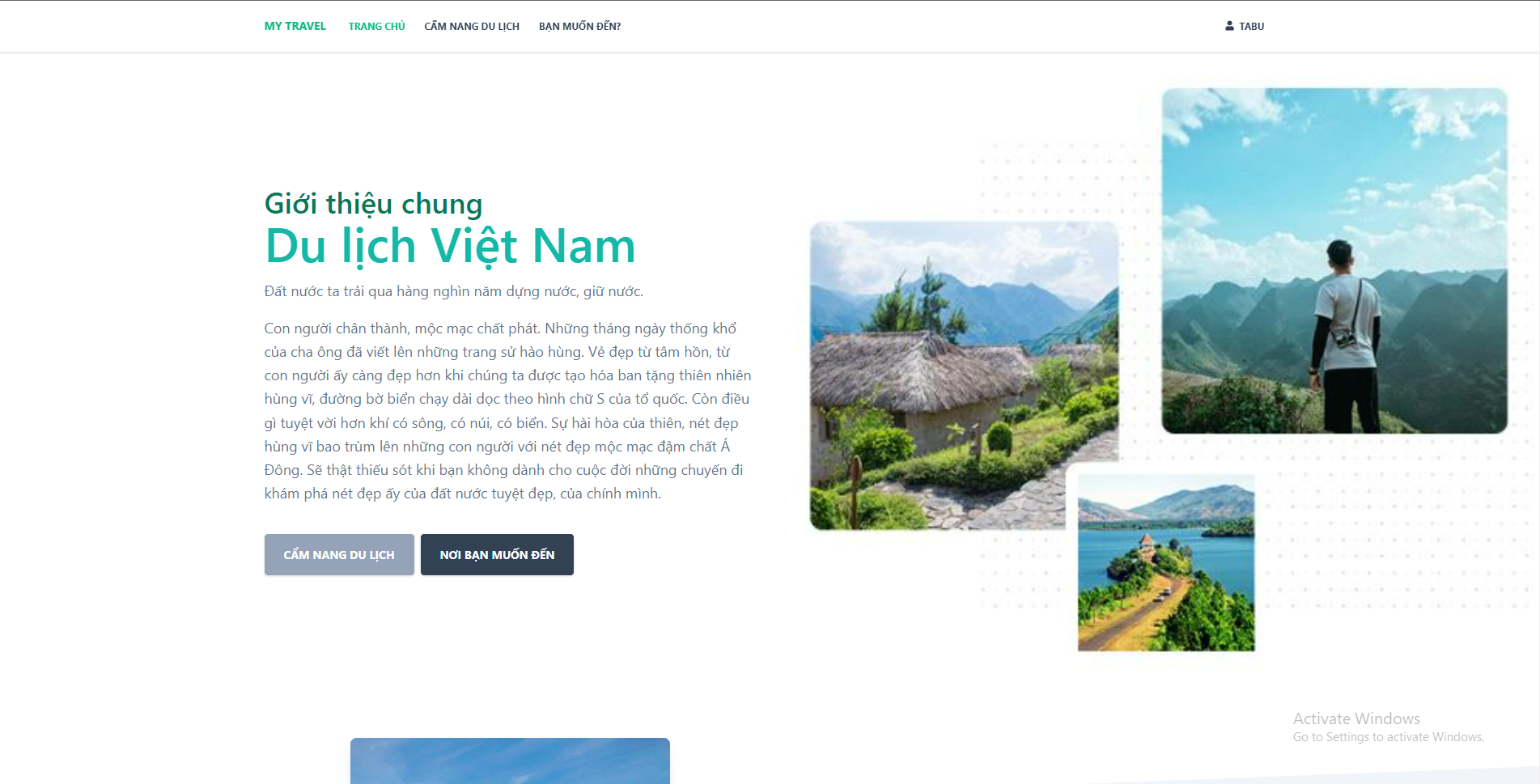
Kết quả đồ án bao gồm: một server, một web client. Triển khai trên trang http://localhost:3001/. Server là nơi xử lý các nghiệp vụ mà phía client yêu cầu, dữ liệu được đóng gói thành định dạng JSON. Client là nơi người dùng tạo những tương 51 tác với hệ thống, được đóng gói bởi Webpack [5]. Client và server giao tiếp với nhau thông qua các API.

1. ***Minh họa các chức năng chính***



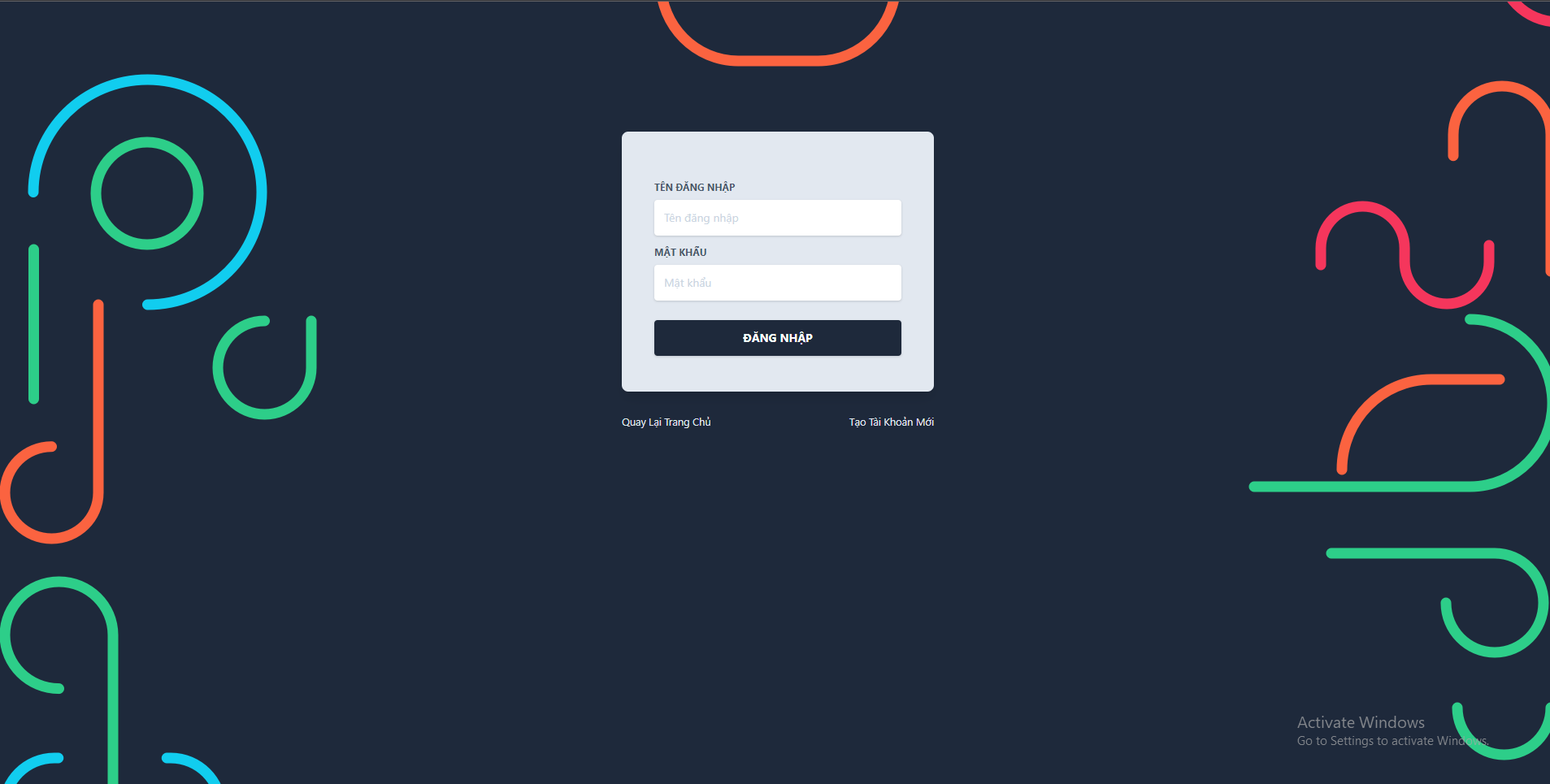
**Hình 16** Màn hình trang chủ khi chưa đăng nhập.

**Hình 15** là màn hình trang chủ khi user chưa có tài khoản để đăng nhập hoặc chưa đăng nhập vẫn có thể truy cập để xem các thông tin từ trang web.



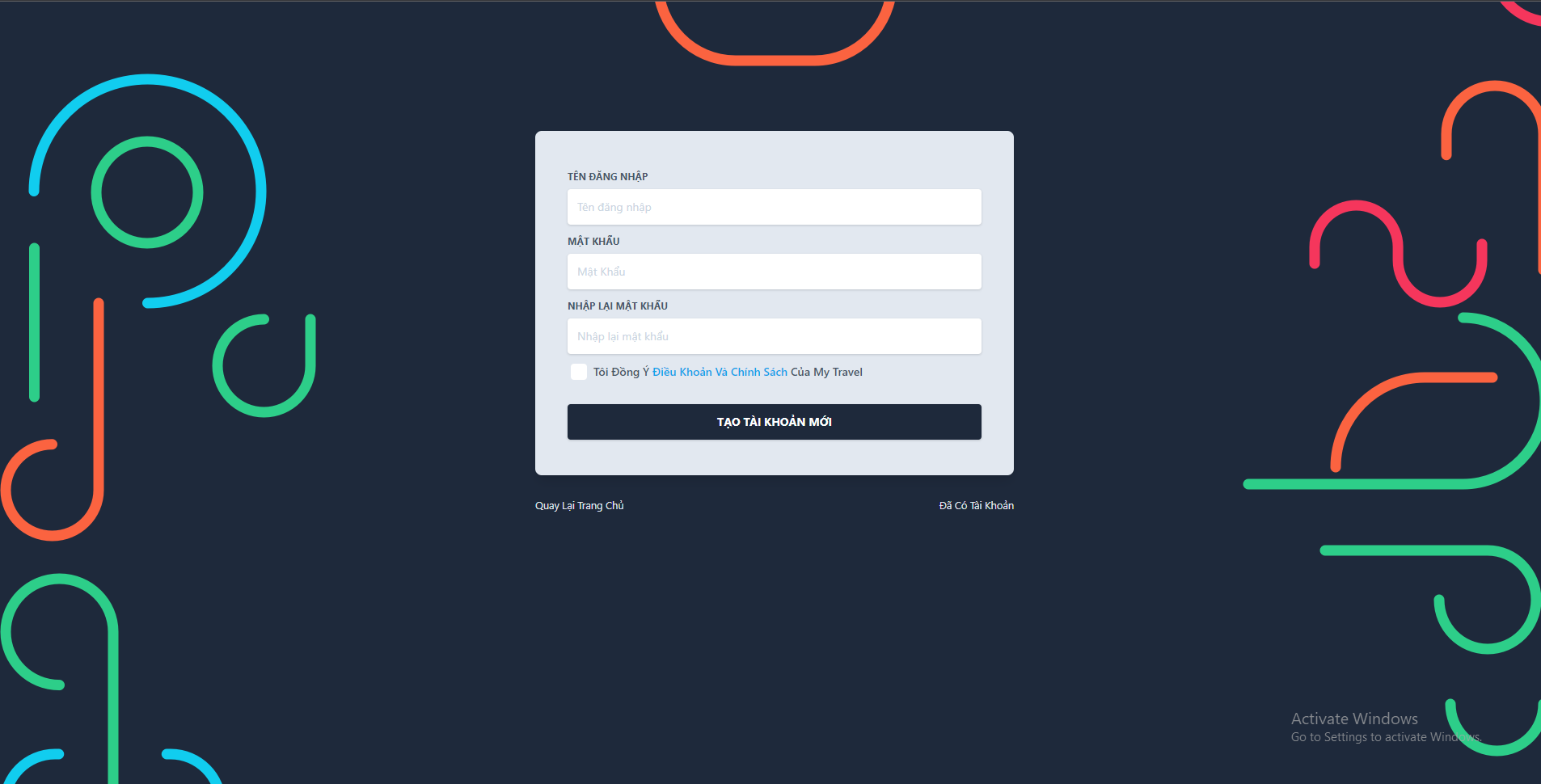
**Hình 17** Màn hình trang chủ khi đã đăng nhập.

**Hình 16** là màn hình trang chủ khi người dùng đã có tài khoản và đăng nhập để xem các thông tin của website. Nhưng đối với người dùng đã có tài khoản thì sẽ có các chức năng: tạo mới bìa viết, thích bài viết, bình luận và đánh giá bài viết trên web.



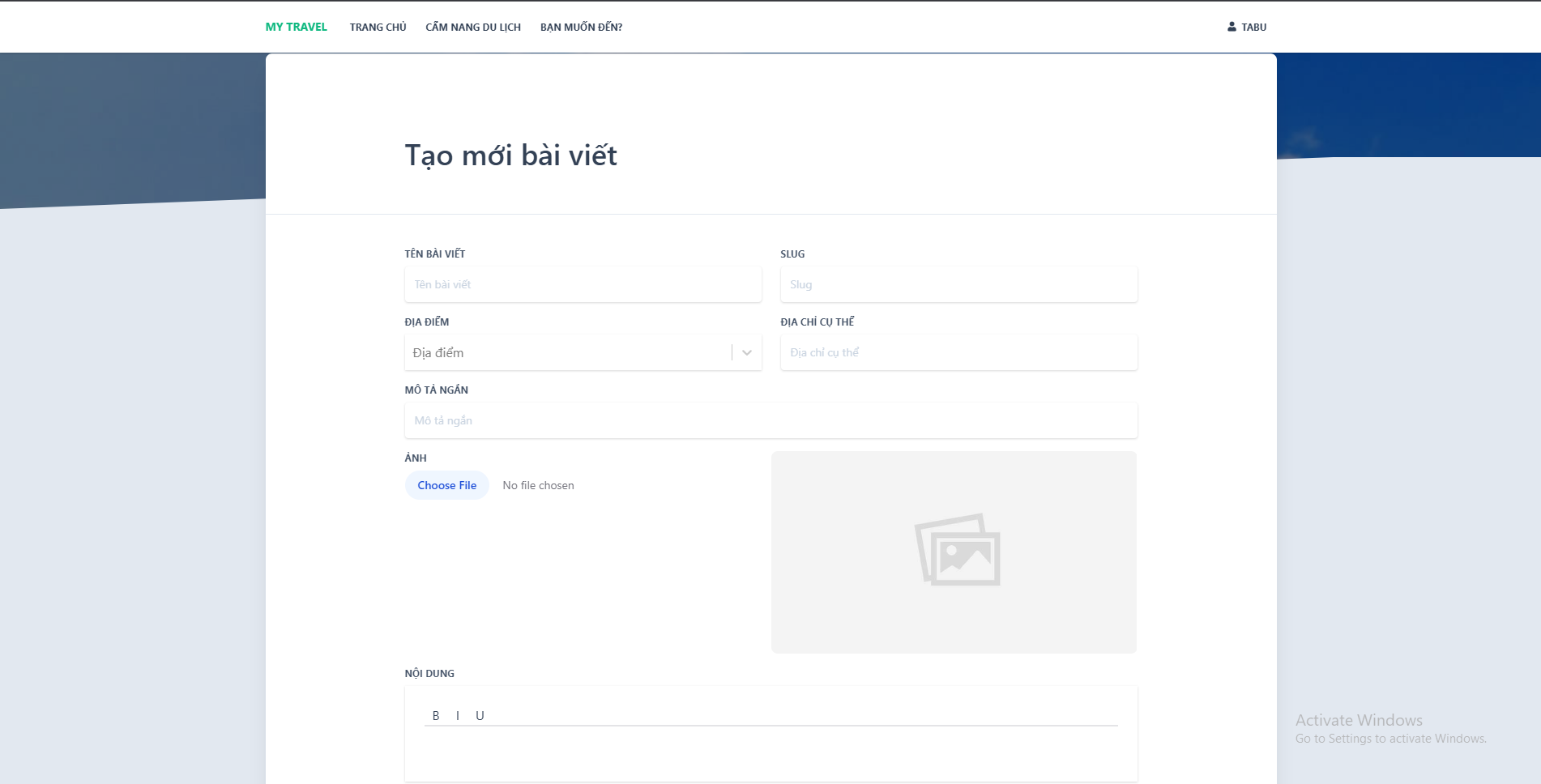
**Hình 18** Màn hình đăng nhập.

**Hình 17** là màn hình đăng nhập vào website đối với người dùng đã có tài khoản thì sẽ vào màn hình này để đăng nhập.



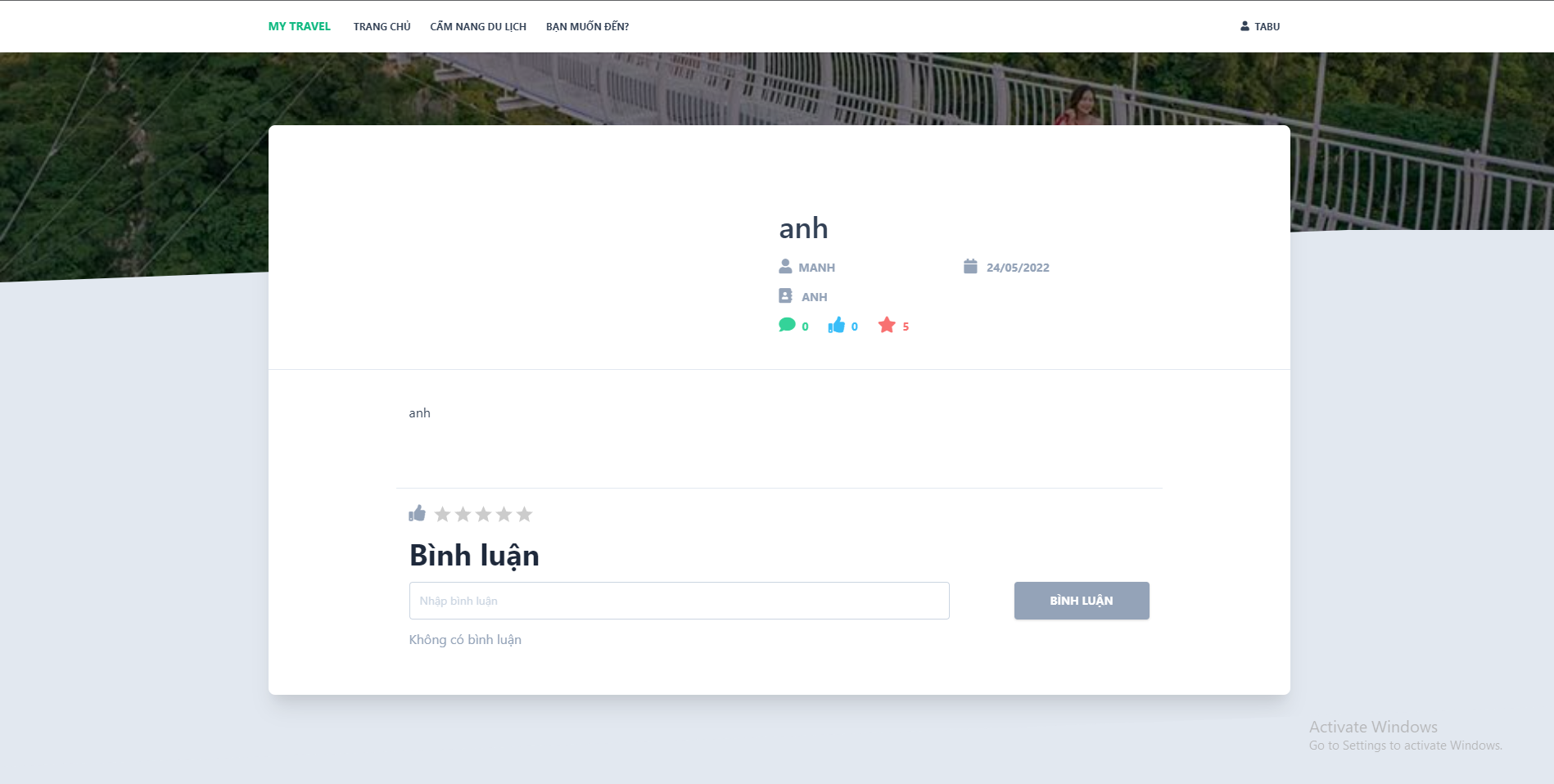
**Hình 19** Màn hình đăng ký tài khoản.

**Hình 18** là màn hình đăng ký. Đối với người dùng chưa có tài khoản để đăng nhập vào website thì sẽ vào màn hình này để tạo tài khoản và sau khi có tài khoản sẽ đăng nhập website và thao tác các chức năng vốn có.



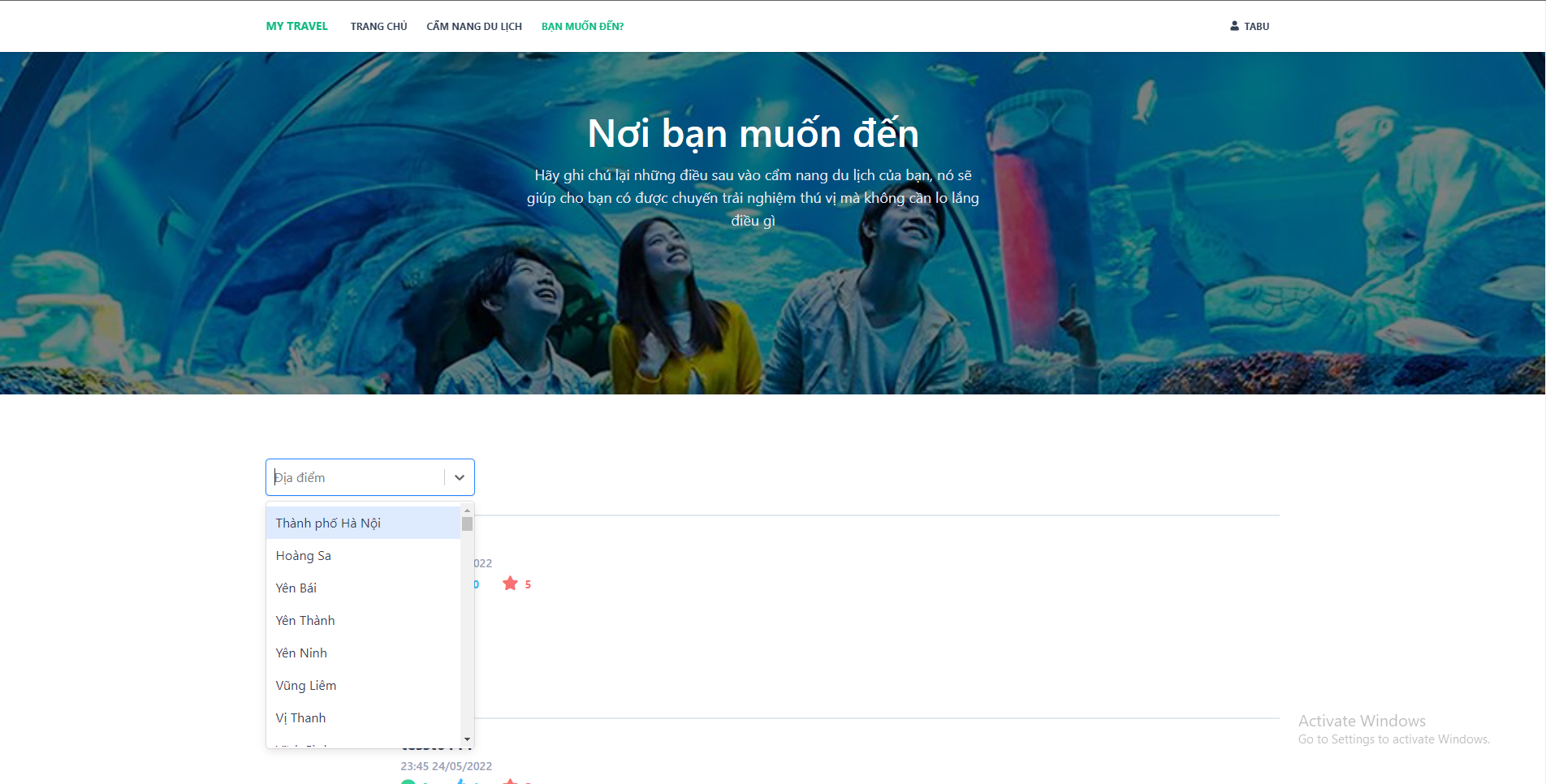
**Hình 20** Màn hình tạo mới bài viết.

**Hình 19** là màn hình tạo mới bài viết. Với người dùng đã có tài khoản và sau khi đăng nhập sẽ có chức năng tạo mới bài biết. Ở đây người dùng có thể đăng tải bài viết với các nội dung và hình ảnh muốn chia sẻ, khi bài viết được tạo thì sẽ được gửi đến để quản trị viên xét duyệt nếu bài viết không vi phạm nội dung hay hình ảnh thì sẽ được duyệt hiển thị trên web và ngược lại sẽ không được hiển thị.



**Hình 21** Màn hình bình luận, thích hoặc đánh giá bài viết.

**Hình 20** là màn hình bình luận, thích hoặc đánh giá bài viết. Tại đây đối với người dùng đã đăng nhập tài khoản thì sẽ có những hoạt động tương tác với bài viết. Phụ thuộc vào nội dung bài viết thì các người đọc sẽ đánh giá hoặc thích/ không thích, bình luận theo quan điểm của mình để người viết có thể nhận đóng góp và chỉnh sửa cho phù hợp hơn.



**Hình 22** Màn hình lọc nơi bạn muốn đến.

**Hình 21** là màn hình lọc nơi bạn muốn đến. Ở đây người dùng có thể lọc theo Filter như trên hình nơi mà mình muốn đến để tìm kiếm thông tin của nơi đó. Khi chọn một trong những địa điểm có trong filter thì danh sách những bài viết về nơi đó sẽ hiển thị để người dùng có thể đọc và tham khảo về thông tin đó.

1. **Kiểm thử**

Dùng công cụ kiểm thử **Selenium IDE** để chạy chương trình kiểm thử.

**Giới thiệu về Selenium:**

* Selenium là công cụ kiểm thử tự động miễn phí (mã nguồn mở) dành cho các ứng dụng web trên các trình duyệt và nền tảng khác nhau.
* Selenium tập trung vào việc tự động hoá các ứng dụng dựa trên web. Kiểm thử được thực hiện bằng cách sử dụng công cụ Selenium thường được gọi là Kiểm thử Selenium.
* Selenium không chỉ là một công cụ mà còn là một bộ phần mềm, mỗi bộ đều đáp ứng nhu cầu kiểm tra khác nhau của một tổ chức.
* Nó gồm bốn thành phần bên dưới:
* Selenium Integrated Development Environment (IDE)
* Selenium Remote Control (RC)
* WebDriver
* Selenium Grid

Sau đây là một số các testcase được tiến hành và kết quả kiểm thử thành công 100%:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Testcase** | **Điều kiện** | **Các bước** | **Kết quả mong muốn** | **Kết quả** |
| Đăng ký với tên đăng nhập chưa tồn tại | Người dùng chưa có tài khoản và tên đăng nhập không bị trùng, mật khẩu hợp lệ. | 1. Nhập tên đăng nhập chưa tồn tại  2. Nhập mật khẩu và Nhập lại mật khẩu hợp lệ  3. Nhấn nút Đăng Ký | Đăng ký thành công | Thành công |
| Đăng ký với tên đăng nhập đã tồn tại | Người dùng thực hiện đăng ký với tên đăng nhập đã tồn tại | 1. Nhập tên đăng nhập đã tồn tại  2. Nhập mật khẩu và Nhập lại mật khẩu hợp lệ  3. Nhấn nút Đăng Ký | Hiển thị thông báo lỗi tại trường Tên Đăng Nhập: “Tên đăng nhập đã tồn tại” | Thành công |
| Đăng nhập với tên đăng nhập và mật khẩu hợp lệ | Người dùng thực hiện đăng nhập khi đã có tài khoản, nhập vào tên đăng nhập và mật khẩu hợp lệ | 1. Nhập tên đăng nhập hợp lệ.  2. Nhập mật khẩu hợp lệ  3. Nhấn nút Đăng Nhập | Đăng nhập thành công và di chuyển tới màn hình trang chủ của website | Thành công |
| Đăng nhập với tên đăng nhập hoặc mật khẩu không hợp lệ | Người dùng nhập tên đăng nhập hoặc mật khẩu không hợp lệ | 1. Nhập tên đăng nhập đúng  2. Nhập sai mật khẩu  3. Nhấn nút đăng nhập | Hiển thị thông báo lỗi về trường mật khẩu | Thành công |
| Tạo bài viết | Người dùng đã có tài khoản truy cập website | 1. Nhấn nút tạo bài viết  2. Nhập thông tin hợp lệ vào các trường: Tên bài viết, Slug, Địa điểm, Địa chỉ cụ thể, Mô tả ngắn, Ảnh, Nội dung | Tạo bài viết thành công | Thành công |

**Bảng 13** Danh sách các testcase mẫu.

1. **Triển khai**

Triển khai chạy test bằng công cụ selenium , tốc độ load nhanh và đưa ra kết quả chính xác cho từng case. Quá trình kiểm thử được diễn ra trên trình duyệt Chrome:

* Mở tool Selenium trên trình duyệt mở rộng.
* Chọn tạo mới project 🡪 tạo mới testcase.
* Thao tác các bước của testcase trên website và selenium ghi lại rồi hiển thị test case trên bảng testcase.
* Chạy lại testcase cho ra kết quả chính xác.

# Chương 5: Các giải pháp và đóng góp nổi bật

1. **Cải thiện cấu trúc Font-end để tăng tốc website**

**- Thu gọn các HTTP request:**

Theo nghiên cứu 80% thời gian load của một trang Web dành cho việc download các phần khác nhau của trang: ảnh, stylesheets, scripts, Flash... Đối với mỗi phần này, một HTTP request sẽ được tạo ra. Vì vậy, trang web càng có nhiều thành phần, thời gian để render trang càng kéo dài.

Đối với trường hợp này, cách nhanh nhất để tăng tốc trang web của bạn đó là tối giản các design.

* Tính toán hợp lý các số lượng các thành phần trên trang web.
* Sử dụng css thay cho images nếu có thể.
* Gộp nhiều stylesheet lại làm một
* Giảm số scripts và đặt chúng ở cuối trang.

Đối với trang web, càng đơn giản càng hiệu quả. Hãy thử giảm số lượng các thành phần trên trang web của mình, bằng việc này bạn đã giảm số lượng các HTTP request cần phải sử dụng để render trang web. Bạn sẽ nhận ra hiệu suất trang web được tăng một cách rõ rệt.

**- Sử dụng cache của trình duyệt**

Bằng việc sử dụng cache của trình duyệt, mỗi khi người dùng ghé thăm trang web của bạn, các thành phần của trang web sẽ được lưu lại trong ổ cứng của người dùng đó, bên trong bộ nhớ cache hoặc bộ nhớ tạm thời, nhờ đó trong lần tiếp theo người dùng đó quay lại trang web, trình duyệt của anh ta có thể load trang và bỏ qua việc phải gửi các HTTP request để lấy các thành phần đã được lưu lại từ trước đó trong cache.

Trong một bài test về cache cho trình duyệt, trong lần đầu tiên người dùng mở trang web của bạn, họ phải downnload các HTML document, stylesheet, javascript file và image trước khi có thể sử dụng trang web đó. Số lượng các thành phần đó vào khoảng 30 (có thể tuỳ vào trang web) và tốn khoảng thời gian là 2.4 giây.

Một khi trang web đã được load và các thành phần được lưu lại trong cache của người dùng, trong lần ghé thăm tiếp theo chỉ một số lượng nhỏ các thành phần cần thiết phải download.

**- Tối ưu hóa các Images**

Đối với các images, bạn cần tập trung vào ba thứ sau: kích thước, format và src attributes.

* Kích thước ảnh
* Định dạng ảnh
* Src attribute

**- Ưu tiên load những nội dung phía trên**

**- Giảm số lượng các plugin sử dụng trong trang web**

1. **Chức năng tạo bài viết chia sẻ kinh nghiệm du lịch**
2. ***Giới thiệu chức năng***

Dựa vào thị hiếu của người dùng, nếu khi chuẩn bị có một chuyến đi họ sẽ tìm kiếm các review trên mạng để tìm hiểu về địa hình, phong tục, lịch trình và đặc điểm thời tiết ở nơi đó để sẵn sàng cho chuyến đi. Để có thể đáp ứng nhu cầu đó, em đã xây dựng hệ thống website để mọi người có thể chia sẻ kinh nghiệm du lịch cho nhau thông qua những bài viết review về địa điểm du lịch hỗ trợ người dùng có thể dễ dàng tìm kiếm được những thông tin mong muốn.

1. ***Các hướng tiếp cận***

* Tìm hiểu thị trường về những địa điểm nổi bật mà các dân du lịch có hứng thú sẽ được hiển thị trên phần lọc để tìm kiếm những thông tin liên quan cho chuyến đi.
* Để có hướng phát triển cho chức năng này cần nghiên cứu thứ nhất về nghiệp vụ và thứ hai là về bài toán để làm thế nào cho tối ưu và thân thiện với người dùng nhất. Em đã đưa ra chức năng gợi ý tìm kiếm bằng cách lọc các thông tin theo địa điểm du lịch để người dùng có thể dễ dàng tìm kiếm và tiếp nhận thông tin. Các các hiển thị thông tin tới người dùng khá quen thuộc thông qua bài viết, bình luận của mọi người như các trang mạng xã hội Facebook, Zalo, Instagram,…v.v. Từ đó người dùng có thể thao thác mà không cần phải hướng dẫn sử dụng quá nhiều.

1. ***Giải pháp***
   * 1. *Thu thập dữ liệu*

Trên các diễn đàn du lịch và các trang mạng xã hội hiện nay nổi bật về trào lưu du lịch. Dựa vào đó để tìm hiểu về xu hướng cũng như thị hiếu của người dùng để đưa ra những thông tin cần thiết và hữu ích cho họ. Cụ thể, hiện nay những vùng đất nổi tiếng du lịch như: Phú quốc, Quy nhơn, Đã nẵng, Đà lạt, Sapa, Hà giang,…v.v. Chúng ta có thể thu thập những thông tin đặc biệt của những địa điểm đó để tạo bài viết giới thiệu cho mọi người biết.

* + 1. *Lọc thông tin theo các địa điểm nổi bật*

Sau khi thu thập được danh sách phim người dùng đã tương tác và điểm tương tác với phim đó. Ta bắt đầu quá trình phân tích dữ liệu, trích xuất thông tin về các địa điểm nổi bật và phổ biến: Phú quốc, Quy nhơn, Đã nẵng, Đà lạt, Sapa, Hà giang,…v.v. Từ đó chọn lọc những thông tin chính xác để cập nhật đưa đến tay người dùng và cũng dùng nguồn thông tin đó để phê duyệt các bài viết được đăng tải nhằm xác thực và đưa ra thông tin hữu ích chân thật nhất.

Ta sử dụng moogos kết nối với cơ sở dữ liệu mongo lấy được list dữ liệu. sau khi lấy được dữ liệu từ bảng post, chung ta dùng map chạy qua một lườt dữ liệu, tại mỗi bản ghi, ta sử dụng công thức tính 2 điểm trên đồ thị (latitude, longitude) xác định khoảng cách từ vị trí của người dùng đến với vị trí của bài viết, sau đó sắp xếp lại dữ liệu theo khoảng cách gần nhất và trả dữ liệu cho người dùng.

1. ***Kết quả thu được***

Sauk hi phân tích và đưa ra những địa điểm nổi bật để người dùng có thể lọc thì từ đó người dùng có thể nhìn thấy 2 chức năng về công cụ này. Thứ nhất, người dùng chưa biết đến địa điểm đó thì có thể tìm kiếm thông tin thông qua việc lọc. Thứ hai, người dùng đã trải nghiệm và biết đến địa điểm đó thì có thể tìm kiếm và để lại những kinh nghiệm sẵn có của mình cho người đi sau.

Khi số lượng người dùng tăng lên, đồng nghĩa với việc lượng thông tin cung cấp cũng sẽ lớn và đáp ứng được nhu cầu tìm kiếm của người dùng. Và việc thu thập dữ liệu để lọc sẽ giúp chọn lọc thông tin và đảm bảo tính ổn định chắc chắn.

# Chương 6: Kết luận và hương phát triển

1. **Kết luận**

Qua thời gian làm đồ án tốt nghiệp, em đã tìm hiểu và học hỏi được rất nhiều kiến thức về xây dựng website. Từ đó chúng em đã thiết lập được cơ sở dữ liệu cho trang web bán hàng trực tuyến qua mạng. Xây dựng giao diện sử dụng React kết hợp Redux, thiết kế các server cung cấp API sử dụng ExpressJS trong NodeJS v.v. Đồng thời có những kiến thức về CSDL, DB, vẽ biểu đồ, thiết kế hệ thống.

Về website chia sẻ kinh nghiệm du lịch em đã hoàn thành các quá trình phân tích thiết kế ứng dụng. Triển khai ứng dụng trên nền tảng web với các luồng hoạt động chính ổn định. Các luồng hoạt động tuy vẫn còn chưa đa dạng nhưng hiện tại luồng chạy đều không có vấn đề. Ví dụ: Đăng nhập, Đăng ký, Quản lý bài viết bởi Admin, Tạo bài viết với người dùng đã có tài khoản.

1. **Hướng phát triển**

* Kiểm soát lỗi có thể phát sinh khi đưa vào sử dụng cho người dùng trên môi trường thực tế.
* Thêm các chức năng đa dạng hơn, ví dụ: cập nhật dự báo thời tiết sắp tới cho các nơi mà người dùng muốn đến, chú thích những lưu ý cụ thể của từng địa điểm để người dùng có thể dễ dàng chuẩn bị cho chuyến đi của mình …v.v
* Kiểm soát chặt chẽ hơn về phần bảo mật định danh người dùng, khi người dùng đăng ký tài khoản cần được gửi tới quản trị viên để phê duyệt cho trạng thái hoạt động hoặc không hoạt động. Điều đó có ích khi có thể báo cáo các tài khoản bị báo cáo và quản trị viên có quyền set trạng thái củ tài khoản đó không được hoạt động.
* Thêm biểu đồ thống kê để quản trị viên nhìn rõ được mức độ tăng trưởng người dùng sử dụng website theo từng tuần hoặc tháng.
* Cập nhật chức năng tạo bài viết có thể đăng video khi người dùng muốn chia sẽ những thước phim để mọi người có thể chiêm ngưỡng.
* Cải thiện về hiệu năng sử dụng website với mức truy cập cao mà web vẫn hoạt động ổn định.

# Tài liệu tham khảo

1. **ReactJS:**

* <https://itviec.com/blog/cau-hoi-phong-van-reactjs-la-gi/?utm_source=google&utm_medium=dis_cpc&utm_campaign=hn_performancemax&gclid=Cj0KCQjw8O-VBhCpARIsACMvVLPSwioVIaWnDXM7I4GHfGaR-4_oo2ZyKr3B81RsNsQA0qyrLlpsNMIaAk6fEALw_wcB>
* <https://viblo.asia/p/gioi-thieu-reactjs-Qbq5QvaGKD8>

1. **Mongo DB:**

* <https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGlY3>

1. **Tailwind CSS:**

* <https://viblo.asia/p/tim-hieu-ve-tailwind-css-924lJp6WKPM>

1. **Next.js - The React Framework:**

* <https://nextjs.org>

1. **So sánh Server Side Rendering và Client Side Rendering trong React:**

* <https://viblo.asia/p/so-sanh-server-side-rendering-va-clientside-rendering-trong-react-V3m5WBrylO7>

1. **Redux · A Predictable State Container for JS Apps:**

* <https://redux.js.org/>

1. **REST API Tutorial:**

* <https://www.restapitutorial.com/>