

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2024-2025

Xây dựng website giới thiệu các ngôi chùa nổi tiếng của tỉnh Trà Vinh sử dụng React.js

Giảng viên hướng dẫn:

TS. Nguyễn Nhứt Lam

Sinh viên thực hiện:

Họ tên: Lê Khánh Đăng

MSSV: 110122047

Lớp: DA22TTA

Trà Vinh, tháng 12 năm 2024

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2024-2025

Xây dựng website giới thiệu các ngôi chùa nổi tiếng của tỉnh Trà Vinh sử dụng React.js

Giảng viên hướng dẫn:

TS. Nguyễn Nhứt Lam

Sinh viên thực hiện:

Họ tên: Lê Khánh Đăng

MSSV: 110122047

Lớp: DA22TTA

Trà Vinh, tháng 12 năm 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

[illegible]

Trà Vinh, ngày tháng năm

Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

[illegible]

Trà Vinh, ngày tháng năm

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn chân thành đến quý thầy cô, các bạn bè đã luôn quan tâm, giúp đỡ em trong suốt quá trình học tập và thực hiện đề tài "Xây dựng website giới thiệu các ngôi chùa nổi tiếng của tỉnh Trà Vinh sử dụng React.js".

Đề tài này là một thử thách lớn đối với em, đòi hỏi em phải có kiến thức chuyên môn vững vàng và kỹ năng lập trình thành thạo. Tuy nhiên, nhờ sự hướng dẫn tận tình của thầy em đã có thể hoàn thành đề tài đúng tiến độ và đạt được kết quả tốt.

Trong quá trình thực hiện đồ án, thầy luôn sát cánh theo dõi từng bước tiến và đưa ra những góp ý kịp thời, giúp em hoàn thành đề tài đúng với yêu cầu đề ra. Em hy vọng sẽ tiếp tục nhận được sự hỗ trợ và chỉ dẫn của thầy trong những dự án và chặng đường học tập sắp tới. Một lần nữa, em xin chân thành cảm ơn và kính chúc thầy luôn mạnh khỏe, hạnh phúc và đạt nhiều thành công trong sự nghiệp giáo dục.

Em cũng xin gửi lời cảm ơn đến các bạn bè trong lớp đã luôn giúp đỡ và động viên em trong suốt quá trình học tập và thực hiện đề tài. Sự giúp đỡ của các bạn đã giúp em có thêm động lực để hoàn thành đề tài một cách tốt nhất.

Do khả năng cũng như kiến thức chuyên môn của em còn hạn chế, nên bài báo cáo không tránh khỏi những sai sót. Em rất mong nhận được ý kiến đóng góp từ quý thầy, cô để có thể khắc phục những thiếu sót và hoàn thiện bài báo cáo hơn trong tương lai. Em cũng xin kính chúc quý thầy, cô và ban lãnh đạo nhà trường luôn dồi dào sức khỏe và thành công trong công việc.

Em xin chân thành cảm ơn!

Trân trọng

MỤC LỤC

TÓM TẮT.....	7
MỞ ĐẦU	9
CHƯƠNG 1: TỔNG QUAN	10
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT	11
2.1. Cấu trúc Web API	11
2.2. Frontend với ReactJS	11
2.2.1. Khái niệm cơ bản về React	11
2.2.2. Các đặc điểm chính của ReactJS	12
2.2.3. Component trong React	12
2.2.4. JSX (JavaScript XML).....	13
2.2.5. State và Props	13
2.2.6. Virtual DOM và Reconciliation	14
2.2.7. Hooks trong React	15
2.2.8. React Router	15
2.2.9. State Management trong React.....	16
2.3. Backend với Node.js	16
2.3.1. Khái niệm cơ bản về Node.js.....	16
2.3.2. Giới thiệu về Express.js	17
2.3.3. Kết hợp giữa Node.js và Express.js	18
2.3.4. Ứng dụng thực tế của Node.js và Express.js	19
2.4. Xác thực JSON Web Token (JWT)	19
2.4.1. Khái niệm về JSON Web Token (JWT).....	19
2.4.2. Cấu trúc của JWT	19
2.4.3. Quá trình xác thực với JWT	20
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU	22
3.1. Đặc tả yêu cầu hệ thống	22
3.1.1. Yêu cầu chức năng.....	22
3.1.2. Yêu cầu phi chức năng.....	23
3.2. Thiết kế dữ liệu	23
3.2.1. Mô hình dữ liệu của chùa trong MongoDB.....	23
3.2.2. Mô hình dữ liệu người dùng trong MongoDB	24
3.2.3. Chi tiết các thực thể	24
3.3. Thiết kế, kiến trúc ứng dụng	25
3.3.1. Kiến trúc toàn cảnh.....	25

3.3.2. Cách frontend nhận dữ liệu.....	25
3.3.3. Thiết kế API.....	25
3.3.4. Thiết kế frontend.....	26
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU.....	29
4.1. Kết quả đạt được	29
4.1.1. Hoàn thiện chức năng chính:	29
4.1.2. Tối ưu hóa hiệu suất:	29
4.1.3. Bảo mật và quản lý dữ liệu:	29
4.2. Giao diện chức năng.....	29
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	36
5.1. Những kết quả đạt được	36
5.2. Hướng phát triển	37
DANH MỤC TÀI LIỆU THAM KHẢO	38

DANH MỤC HÌNH ẢNH

Hình 2.1 Sơ đồ hoạt động của Web API.....	11
Hình 2.2 Ví dụ Functional components	12
Hình 2.3 Ví dụ Class components	13
Hình 2.4 Ví dụ JavaScript XML.....	13
Hình 2.5 Biên dịch thành JavaScript.....	13
Hình 2.6 Ví dụ sử dụng useState (hook)	14
Hình 2.7 Ví dụ Props	14
Hình 2.8 Ví dụ về useEffect	15
Hình 2.9 Ví dụ về React Router	15
Hình 2.10 Ví dụ về một ứng dụng Express cơ bản	18
Hình 2.11 Ví dụ tạo JWT trong Node.js	20
Hình 2.12 Ví dụ xác thực JWT trong Node.js.....	21
Hình 3.1 Sơ đồ cấu trúc website.....	22
Hình 3.2 Mô hình dữ liệu của chùa trong MongoDB	23
Hình 3.3 mô hình dữ liệu người dùng trong MongoDB	24
Hình 3.4 Cấu trúc thư mục frontend.....	26
Hình 3.5 Sơ đồ cấu trúc thư mục và chức năng	26
Hình 3.6 Tập tin App.js	27
Hình 4.1 Trang chủ.....	30
Hình 4.2 Trang thông tin chi tiết về chùa.....	30
Hình 4.3 Trang thông tin chi tiết bản đồ chùa.....	31
Hình 4.4 Trang quản lý chùa	31
Hình 4.5 Trang quản lý chùa	32
Hình 4.6 Thông báo xác nhận khi thêm chùa.....	32
Hình 4.7 Thông báo xác nhận khi xóa chùa	32
Hình 4.8 Thông báo xác nhận khi xóa 1 lịch sử chùa	32
Hình 4.9 Giao diện đăng ký	33
Hình 4.10 Giao diện đăng nhập.....	33
Hình 4.11 Giao diện trang giới thiệu.....	34
Hình 4.12 Giao diện trang giới thiệu.....	34
Hình 4.13 Giao diện trang giới thiệu.....	34
Hình 4.14 Giao diện trang liên hệ	35

TÓM TẮT

Vấn đề nghiên cứu

Tỉnh Trà Vinh sở hữu di sản văn hóa Khmer độc đáo và nhiều ngôi chùa nổi tiếng, tuy nhiên thông tin và hình ảnh về các di sản này chưa được quảng bá rộng rãi trên nền tảng kỹ thuật số. Điều này khiến việc tiếp cận thông tin về văn hóa, lịch sử và du lịch tại Trà Vinh gặp nhiều hạn chế.

Các hướng tiếp cận

Ứng dụng công nghệ số hóa: Nghiên cứu và sử dụng các công cụ web hiện đại như React.js, Node.js và MongoDB để phát triển một hệ thống quản lý thông tin và trình bày dữ liệu về tỉnh Trà Vinh.

Tối ưu hóa trải nghiệm người dùng: Đảm bảo giao diện hiện đại, tương tác động và tương thích với thiết bị di động.

Áp dụng SEO cơ bản: Nghiên cứu các tiêu chuẩn SEO để giúp website dễ dàng được tìm thấy trên các công cụ tìm kiếm, qua đó tiếp cận người dùng hiệu quả hơn.

Cách giải quyết vấn đề

1. Phân tích và thu thập dữ liệu:

Nghiên cứu về lịch sử, văn hóa của tỉnh Trà Vinh và các ngôi chùa Khmer nổi tiếng.

Thu thập tài liệu, hình ảnh và video minh họa.

2. Phát triển hệ thống:

Frontend: Sử dụng React.js để tạo giao diện tương tác động và Tailwind CSS để tối ưu hiển thị trên mọi thiết bị.

Backend: Sử dụng Node.js với Express.js để xây dựng API và MongoDB để lưu trữ dữ liệu.

SEO và hiệu suất: Tối ưu hóa các yếu tố như metadata, sitemap, và phân trang để cải thiện khả năng truy cập và tốc độ tải trang.

3. Kiểm thử và triển khai:

Kiểm thử hệ thống trên nhiều trình duyệt và thiết bị khác nhau.

Triển khai website lên môi trường thực tế với hosting hoặc nền tảng đám mây như AWS/Heroku.

4. Kết quả đạt được

Website hoàn thiện: Hệ thống hoạt động ổn định, cung cấp thông tin chi tiết và

chính xác về tỉnh Trà Vinh, các ngôi chùa Khmer.

Tối ưu hiệu suất: Đạt điểm tối ưu trên các công cụ kiểm tra SEO và hiệu năng như Google Lighthouse.

Trải nghiệm người dùng: Giao diện thu hút, dễ sử dụng và tương thích tốt với các thiết bị di động.

Quảng bá văn hóa: Góp phần đưa hình ảnh tỉnh Trà Vinh đến gần hơn với cộng đồng trong và ngoài nước.

MỞ ĐẦU

Tỉnh Trà Vinh là một địa danh đặc biệt của đồng bằng sông Cửu Long, nổi tiếng với những ngôi chùa Khmer độc đáo và bản sắc văn hóa truyền thống. Việc xây dựng website sử dụng ReactJS sẽ giúp nâng cao trải nghiệm người dùng, góp phần bảo tồn và quảng bá văn hóa thu hút khách du lịch.

Mục đích nghiên cứu phát triển một website giúp du khách và người dân tìm hiểu lịch sử, văn hóa Trà Vinh và các ngôi chùa Khmer. Cung cấp thông tin chính xác, dễ dàng truy cập, đồng thời quảng bá hình ảnh địa phương trên nền tảng số.

Đối tượng nghiên cứu:

Các ngôi chùa nổi tiếng tại tỉnh Trà Vinh: Đây là đối tượng chính trong dự án. Bao gồm các ngôi chùa có giá trị lịch sử, văn hóa, và tôn giáo, có ảnh hưởng lớn đến đời sống tinh thần của người dân địa phương và khách du lịch.

Lịch sử các ngôi chùa: Tìm hiểu lịch sử, quá trình hình thành, phát triển, và các sự kiện quan trọng liên quan đến các ngôi chùa nổi tiếng tại Trà Vinh.

Tỉnh Trà Vinh: Tìm hiểu về tỉnh Trà Vinh nói chung, bao gồm các thông tin về địa lý, văn hóa, và các điểm tham quan nổi bật khác, nhằm cung cấp bối cảnh rộng hơn cho các ngôi chùa.

Phạm vi nghiên cứu:

Địa lý: Phạm vi nghiên cứu bao gồm tất cả các ngôi chùa nổi tiếng nằm trong địa bàn tỉnh Trà Vinh. Các ngôi chùa này có thể là các chùa cổ, chùa mới, hoặc các chùa có ảnh hưởng lớn đến cộng đồng.

Lịch sử và Văn hóa: Nghiên cứu lịch sử các ngôi chùa, sự kiện quan trọng trong quá trình phát triển của từng ngôi chùa và vai trò của chúng trong cộng đồng Phật giáo và đời sống văn hóa của người dân Trà Vinh.

Phạm vi nội dung: Các thông tin liên quan đến ngôi chùa sẽ bao gồm: lịch sử hình thành, đặc điểm kiến trúc, các nghi thức tôn giáo, các sự kiện nổi bật, hình ảnh và video về ngôi chùa.

Nghiên cứu về công nghệ và phát triển website: Nghiên cứu về các công nghệ phát triển web hiện đại như React.js, Node.js, MongoDB, và các phương pháp tối ưu hóa SEO, thiết kế giao diện người dùng và tối ưu hóa trên thiết bị di động.

CHƯƠNG 1: TỔNG QUAN

Trà Vinh là một tỉnh nằm ở khu vực Đồng bằng sông Cửu Long, Việt Nam, nổi bật với nền văn hóa đặc sắc và sự đa dạng về tôn giáo. Trong đó, Phật giáo đóng vai trò quan trọng và các ngôi chùa là trung tâm tôn giáo, văn hóa của cộng đồng. Các ngôi chùa không chỉ là nơi sinh hoạt tôn giáo mà còn là những di tích lịch sử và văn hóa có giá trị đặc biệt, phản ánh đời sống tinh thần của người dân Trà Vinh qua nhiều thế hệ.

Tuy nhiên, trong bối cảnh hiện đại hóa và sự phát triển của công nghệ thông tin, việc quảng bá và giới thiệu về các ngôi chùa và lịch sử của chúng tới cộng đồng trong và ngoài tỉnh vẫn còn nhiều hạn chế. Các thông tin về các ngôi chùa, lịch sử và giá trị văn hóa chưa được trình bày một cách hệ thống và dễ tiếp cận đối với đông đảo người dân và khách du lịch.

Website giới thiệu về tỉnh Trà Vinh và các ngôi chùa nổi tiếng tại đây sẽ giúp giải quyết vấn đề này. Nó sẽ cung cấp một nền tảng trực tuyến dễ dàng truy cập, nơi người dùng có thể tìm thấy thông tin về các ngôi chùa, lịch sử, kiến trúc, cũng như các hoạt động tôn giáo và văn hóa diễn ra tại từng ngôi chùa. Mục tiêu chính của dự án là xây dựng một website hiện đại, dễ sử dụng, tối ưu hóa hiệu suất và khả năng tương tác động, đồng thời đảm bảo tính tương thích với các thiết bị di động và tối ưu SEO cơ bản để dễ dàng tiếp cận người dùng.

Cụ thể, website sẽ cung cấp các chức năng như:

Giới thiệu các ngôi chùa nổi tiếng tại Trà Vinh, bao gồm thông tin về lịch sử, địa lý, kiến trúc và các sự kiện đặc biệt.

Cung cấp nền tảng cho phép thêm, sửa, xóa thông tin về các ngôi chùa, giúp website luôn được cập nhật với các dữ liệu mới nhất.

Tối ưu hóa SEO để người dùng dễ dàng tìm kiếm thông tin trên các công cụ tìm kiếm.

Đảm bảo website dễ dàng truy cập trên mọi thiết bị, từ máy tính đến điện thoại di động.

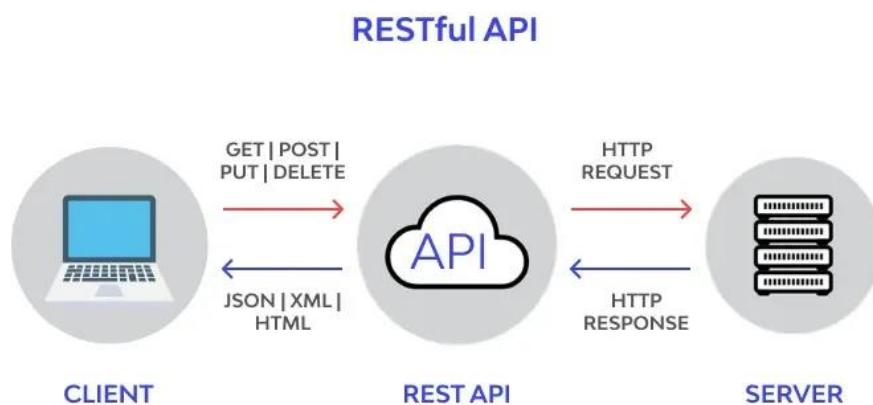
Vấn đề nghiên cứu trong đề tài này sẽ tập trung vào việc thiết kế và phát triển website dựa trên công nghệ hiện đại như React.js cho front-end, Node.js cho back-end, và MongoDB cho cơ sở dữ liệu. Các yêu cầu về giao diện, hiệu suất, SEO và khả năng tương tác động sẽ được giải quyết trong suốt quá trình phát triển hệ thống.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Cấu trúc Web API

Web API (Application Programming Interface) là một giao diện lập trình ứng dụng cho phép các ứng dụng giao tiếp và trao đổi dữ liệu qua mạng internet. Web API thường sử dụng giao thức HTTP hoặc HTTPS để truyền tải dữ liệu và cho phép các ứng dụng khác nhau (có thể là ứng dụng web, ứng dụng di động, hoặc hệ thống backend) kết nối với nhau.

Web API cung cấp một bộ quy tắc, phương thức và định dạng dữ liệu để các hệ thống có thể thực hiện các hành động như tạo mới, lấy, cập nhật hoặc xóa dữ liệu mà không cần phải hiểu rõ về cách thức hoạt động nội bộ của hệ thống mà nó giao tiếp.



Hình 2.1 Sơ đồ hoạt động của Web API

2.2. Frontend với ReactJS

2.2.1. Khái niệm cơ bản về React

ReactJS là một thư viện JavaScript dành cho việc xây dựng giao diện người dùng (UI) trong các ứng dụng web. React được phát triển bởi Facebook và được sử dụng rộng rãi vì khả năng xây dựng giao diện UI mượt mà và hiệu quả.

React được thiết kế với mục tiêu component hóa giao diện người dùng, tức là chia giao diện thành các thành phần nhỏ và có thể tái sử dụng. Điều này giúp việc phát triển, bảo trì và mở rộng ứng dụng trở nên dễ dàng hơn.

2.2.2. Các đặc điểm chính của ReactJS

Component-based architecture: React xây dựng ứng dụng dựa trên các component (thành phần). Mỗi component có thể là một đơn vị giao diện (UI) hoặc một nhóm các thành phần con.

Declarative UI: React sử dụng một cách tiếp cận khai báo (declarative) thay vì imperative. Bạn chỉ cần mô tả giao diện như thế nào dựa trên state của ứng dụng và React sẽ tự động cập nhật UI khi state thay đổi.

Virtual DOM: React sử dụng một Virtual DOM để tối ưu hóa hiệu suất. Thay vì cập nhật trực tiếp DOM, React sẽ tạo một phiên bản ảo của DOM, so sánh với DOM thực tế và chỉ cập nhật những phần thay đổi.

Unidirectional data flow: Dữ liệu trong React chảy theo một hướng duy nhất, từ component cha xuống các component con qua props. Điều này giúp dễ dàng kiểm soát dữ liệu và tránh các lỗi không mong muốn.

2.2.3. Component trong React

React ứng dụng mô hình component để chia nhỏ giao diện thành các phần có thể tái sử dụng.

Functional Component: Đây là loại component đơn giản, thường sử dụng hooks (từ phiên bản React 16.8) để quản lý state và các tác vụ khác.

Class Component: Trước khi có hooks, React sử dụng class components, nơi bạn có thể khai báo state và các phương thức. Tuy nhiên, hiện nay functional components đã trở nên phổ biến hơn nhờ vào sự xuất hiện của hooks.

Functional components

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

Hình 2.2 Ví dụ Functional components

Class components là các class mở rộng từ `React.Component`. Class component có thể chứa state và phương thức lifecycle

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello, {this.props.name}</h1>;  
  }  
}
```

Hình 2.3 Ví dụ Class components

2.2.4. JSX (JavaScript XML)

JSX là một cú pháp đặc biệt cho phép bạn viết mã HTML trong JavaScript. JSX không phải là mã HTML thực sự mà là một cách React mô tả các component UI.

JSX cho phép bạn viết cấu trúc UI theo kiểu giống HTML, nhưng bên dưới, nó sẽ được biên dịch thành `React.createElement()`. Điều này giúp bạn dễ dàng xây dựng giao diện và kết hợp HTML với JavaScript

```
const element = <h1>Hello, world!</h1>;
```

Hình 2.4 Ví dụ JavaScript XML

```
const element = React.createElement('h1', null, 'Hello, world!');
```

Hình 2.5 Biên dịch thành JavaScript

2.2.5. State và Props

- State:

State là một đối tượng được sử dụng để lưu trữ dữ liệu và có thể thay đổi trong thời gian thực. Khi state thay đổi, React sẽ tự động render lại component.

State là dữ liệu "cục bộ", mỗi component có thể quản lý một state riêng.

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0); // Tạo state count với giá trị mặc định 0
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

Hình 2.6 Ví dụ sử dụng useState (hook)

- Props:

Props là các đối tượng mà bạn có thể truyền từ component cha xuống component con. Props được sử dụng để truyền dữ liệu và hành vi giữa các component.

Props là bất biến (immutable), nghĩa là component con không thể thay đổi giá trị của props mà nó nhận được.

```
function Greeting(props) {
  return <h1>Hello, {props.name}</h1>;
}

// Sử dụng component Greeting với props
<Greeting name="John" />
```

Hình 2.7 Ví dụ Props

2.2.6. Virtual DOM và Reconciliation

Virtual DOM là một khái niệm quan trọng trong React. Khi bạn thay đổi state hoặc props trong ứng dụng, React sẽ không ngay lập tức cập nhật DOM thực tế. Thay vào đó, nó sẽ cập nhật một bản sao của DOM trong bộ nhớ (Virtual DOM).

Khi state hoặc props thay đổi, React sẽ:

- Tạo một Virtual DOM mới.
- So sánh Virtual DOM mới với Virtual DOM cũ.
- Chỉ cập nhật những phần của DOM thực tế mà có sự thay đổi.

Quá trình này giúp giảm thiểu số lượng thao tác với DOM thực tế, từ đó tối ưu hiệu suất ứng dụng.

2.2.7. Hooks trong React

Hooks được giới thiệu từ React 16.8, cho phép bạn sử dụng state và các tính năng khác mà không cần phải sử dụng class components.

useState: Hook giúp bạn quản lý state trong component chức năng.

useEffect: Hook cho phép bạn thực hiện các tác vụ phụ (side effects) như lấy dữ liệu từ API hoặc thay đổi DOM sau khi render.

useContext: Hook giúp bạn chia sẻ dữ liệu giữa các component mà không cần phải truyền qua props.

```
import React, { useState, useEffect } from 'react';

function DataFetching() {
  const [data, setData] = useState([]);

  useEffect(() => {
    fetch('https://api.example.com/data')
      .then(response => response.json())
      .then(data => setData(data));
  }, []); // Chạy once khi component mount

  return <ul>{data.map(item => <li key={item.id}>{item.name}</li>)}</ul>;
}
```

Hình 2.8 Ví dụ về useEffect

2.2.8. React Router

React Router là một thư viện dùng để điều hướng trong ứng dụng React. Nó cho phép bạn tạo các route để chuyển hướng người dùng giữa các trang mà không cần tải lại toàn bộ trang web.

```
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

function App() {
  return (
    <Router>
      <nav>
        <Link to="/home">Home</Link>
        <Link to="/about">About</Link>
      </nav>
      <Route path="/home" component={Home} />
      <Route path="/about" component={About} />
    </Router>
  );
}
```

Hình 2.9 Ví dụ về React Router

2.2.9. State Management trong React

Khi ứng dụng React của bạn trở nên lớn hơn, quản lý state trở thành một vấn đề quan trọng. Bạn có thể sử dụng Context API hoặc thư viện bên ngoài như Redux để quản lý state toàn cục.

Context API: Cho phép chia sẻ state giữa các component mà không cần phải truyền props qua nhiều lớp component.

Redux: Thư viện quản lý state toàn cục, sử dụng mô hình action, reducer và store.

2.3. Backend với Node.js

2.3.1. Khái niệm cơ bản về Node.js

Node.js là một nền tảng phần mềm mã nguồn mở, chạy trên JavaScript và được thiết kế để phát triển ứng dụng phía server. Node.js sử dụng cơ chế non-blocking, event-driven architecture, có nghĩa là nó không chặn các tác vụ trong khi đang thực hiện các công việc khác.

- Các đặc điểm chính của Node.js:

Cơ chế I/O không đồng bộ: Node.js sử dụng một hệ thống I/O không đồng bộ, giúp xử lý hàng triệu kết nối đồng thời mà không bị "block". Điều này giúp ứng dụng có thể phục vụ nhiều yêu cầu mà không cần phải tạo thêm thread mới cho mỗi kết nối.

Single Thread (Luồng đơn): Node.js chạy trên một luồng duy nhất (single-threaded event loop). Điều này giúp giảm thiểu việc sử dụng bộ nhớ và cải thiện hiệu suất khi xử lý nhiều kết nối đồng thời.

V8 Engine: Node.js sử dụng V8 engine của Google, giúp JavaScript chạy nhanh và hiệu quả.

Sự kiện và callback: Node.js hoạt động theo mô hình sự kiện (event-driven), nơi mà khi một tác vụ cần được xử lý (ví dụ như đọc file, truy vấn cơ sở dữ liệu), nó sẽ thực hiện một callback khi tác vụ đó hoàn tất.

- Các ứng dụng của Node.js:

Web server: Node.js được sử dụng để xây dựng các server HTTP mạnh mẽ, phục vụ các trang web hoặc API.

Real-time applications: Node.js rất phù hợp cho các ứng dụng yêu cầu kết nối thời gian thực, chẳng hạn như trò chuyện trực tuyến, game online.

Microservices: Vì Node.js có thể xử lý nhiều kết nối đồng thời, nó rất phù hợp cho các ứng dụng microservices.

2.3.2. Giới thiệu về Express.js

Express.js là một framework dành cho Node.js, giúp việc xây dựng các ứng dụng web và API trở nên dễ dàng hơn. Express cung cấp các công cụ để xử lý yêu cầu HTTP, quản lý các route (định tuyến) và cung cấp các middleware để xử lý các tác vụ như xác thực, xử lý lỗi, v.v.

- Các đặc điểm chính của Express.js:

Sử dụng mô hình MVC (Model-View-Controller): Express.js giúp dễ dàng tổ chức mã nguồn theo mô hình MVC, giúp phân tách các phần của ứng dụng.

Routing (Định tuyến): Express cho phép bạn dễ dàng định nghĩa các routes cho các phương thức HTTP (GET, POST, PUT, DELETE). Bạn có thể xử lý các yêu cầu từ client và trả về kết quả.

Middleware: Middleware là các hàm xử lý các yêu cầu trước khi chúng đến route handler. Express cho phép bạn dễ dàng thêm middleware để thực hiện các tác vụ như kiểm tra xác thực, xử lý form data, hoặc quản lý CORS.

Cấu hình dễ dàng: Express cho phép bạn dễ dàng cấu hình ứng dụng, bao gồm thiết lập cổng, template engines và các cấu hình khác như CORS, cookie, body-parser

Xử lý lỗi: Express cung cấp cơ chế middleware để xử lý các lỗi phát sinh trong ứng dụng, giúp ứng dụng trở nên mạnh mẽ và dễ bảo trì.

- Cách hoạt động của Express:

Express.js là một layer thêm vào Node.js, giúp làm việc với HTTP dễ dàng hơn. Cấu trúc cơ bản của một ứng dụng Express bao gồm:

Khởi tạo Express: Tạo một instance của ứng dụng Express.

Định nghĩa routes: Các đường dẫn và phương thức HTTP được định nghĩa cho ứng dụng.

Middleware: Các hàm middleware được sử dụng để xử lý các tác vụ như kiểm tra xác thực, dữ liệu đầu vào, xử lý lỗi, v.v.

Lắng nghe yêu cầu: Lắng nghe các yêu cầu từ client và gửi phản hồi.

```
const express = require('express');
const app = express();

// Định nghĩa route cho phương thức GET
app.get('/', (req, res) => {
  res.send('Hello, world!');
});

// Định nghĩa route cho phương thức POST
app.post('/user', (req, res) => {
  res.send('User Created');
});

// Lắng nghe yêu cầu ở cổng 3000
app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Hình 2.10 Ví dụ về một ứng dụng Express cơ bản

`app.get()`: Định nghĩa route cho yêu cầu GET tại đường dẫn `/`.

`app.post()`: Định nghĩa route cho yêu cầu POST tại đường dẫn `/user`.

`app.listen()`: Lắng nghe các yêu cầu đến cổng 3000.

2.3.3. Kết hợp giữa Node.js và Express.js

Mặc dù Node.js cung cấp các công cụ cơ bản để tạo một ứng dụng web (ví dụ: xử lý HTTP requests và responses), Express.js cung cấp các tính năng nâng cao giúp việc phát triển ứng dụng trở nên dễ dàng và nhanh chóng hơn. Với Express, bạn không cần phải viết mã xử lý HTTP request và response từ đầu mà chỉ cần tập trung vào logic ứng dụng của mình.

- Các lợi ích khi sử dụng Express với Node.js:

Đơn giản hóa việc xử lý routing: Với Express, bạn có thể dễ dàng định nghĩa các route cho các phương thức HTTP khác nhau (GET, POST, PUT, DELETE).

Middleware linh hoạt: Express cho phép bạn sử dụng middleware để xử lý các tác vụ như xác thực, xử lý dữ liệu và nhiều thứ khác trước khi gửi phản hồi về client.

Cấu trúc mã rõ ràng: Express giúp tổ chức mã nguồn tốt hơn, giúp dễ dàng quản lý các phần khác nhau của ứng dụng như routes, controllers và middleware.

Hỗ trợ nhiều template engines: Express hỗ trợ các template engines như Pug, EJS để render HTML động, phù hợp cho các ứng dụng cần giao diện người dùng động.

Mạnh mẽ trong việc xây dựng RESTful API: Express giúp xây dựng các API RESTful mạnh mẽ và dễ dàng với các phương thức HTTP như GET, POST, PUT, DELETE.

2.3.4. Ứng dụng thực tế của Node.js và Express.js

Web Servers: Tạo web server, phục vụ các trang HTML, CSS, JavaScript.

RESTful APIs: Xây dựng các API RESTful cho các ứng dụng di động, ứng dụng web hoặc ứng dụng IoT.

Real-time applications: Xây dựng các ứng dụng yêu cầu kết nối thời gian thực như chat hoặc game online.

Microservices: Node.js và Express rất phù hợp với mô hình microservices vì khả năng xử lý nhiều kết nối đồng thời và hiệu suất cao.

Kết luận Node.js và Express.js là hai công nghệ mạnh mẽ giúp xây dựng các ứng dụng web và API hiệu quả. Node.js cung cấp nền tảng chạy JavaScript phía server, còn Express.js là framework giúp tối ưu hóa quá trình phát triển ứng dụng bằng cách cung cấp các công cụ để xử lý HTTP requests, định tuyến, middleware và nhiều tính năng khác. Khi kết hợp với nhau, chúng giúp xây dựng các ứng dụng web nhanh chóng, mạnh mẽ và dễ bảo trì.

2.4. Xác thực JSON Web Token (JWT)

2.4.1. Khái niệm về JSON Web Token (JWT)

JSON Web Token (JWT) là một tiêu chuẩn mở (RFC 7519) cho phép trao đổi thông tin giữa các bên một cách an toàn dưới dạng đối tượng JSON. Thông tin trong JWT có thể được xác thực và tin cậy nhờ vào việc ký số (digital signature). JWT thường được sử dụng để xác thực người dùng trong các ứng dụng web và API.

2.4.2. Cấu trúc của JWT

JWT có ba phần chính:

Header: Chứa thông tin về thuật toán mã hóa được sử dụng để ký JWT, ví dụ như HMAC SHA256 hoặc RSA.

Payload: Chứa thông tin về người dùng và các dữ liệu mà bạn muốn trao đổi, gọi là claims. Claims có thể là:

- Registered Claims: Các claims tiêu chuẩn đã được định nghĩa như iss (issuer), exp (expiration), sub (subject), v.v.
- Public Claims: Các claims mà bạn có thể tự định nghĩa, miễn là không trùng với các registered claims.
- Private Claims: Các claims mà bạn tự định nghĩa cho ứng dụng của mình.

Signature: Được tạo ra bằng cách ký kết phần Header và Payload với một khóa bí mật (secret key) hoặc khóa công khai (public key) nếu sử dụng RSA hoặc ECDSA. Mục đích là để xác nhận tính toàn vẹn của token và xác thực người gửi.

2.4.3. Quá trình xác thực với JWT

Quá trình xác thực sử dụng JWT có thể chia thành các bước chính sau:

- Tạo JWT:

Bước 1: Người dùng đăng nhập vào hệ thống và cung cấp thông tin đăng nhập (ví dụ: tên người dùng và mật khẩu).

Bước 2: Hệ thống kiểm tra thông tin đăng nhập. Nếu hợp lệ, hệ thống tạo một JWT chứa các thông tin liên quan đến người dùng (ví dụ: ID người dùng, quyền truy cập) và ký nó bằng một khóa bí mật.

Bước 3: JWT được gửi trả về client (thường là dưới dạng một cookie hoặc trong header Authorization).

```
const jwt = require('jsonwebtoken');

// Dữ liệu người dùng để tạo payload
const user = {
  id: 1,
  username: 'exampleUser'
};

// Tạo JWT (sign the token)
const token = jwt.sign(user, 'secretKey', { expiresIn: '1h' });
console.log(token);
```

Hình 2.11 Ví dụ tạo JWT trong Node.js

jwt.sign(): Hàm này dùng để tạo JWT. Nó nhận vào payload (dữ liệu người dùng), khóa bí mật (secretKey) và các tùy chọn như thời gian hết hạn (expiresIn).

- Gửi JWT:

JWT thường được gửi kèm trong header của yêu cầu HTTP, ví dụ trong header Authorization

Authorization: Bearer <JWT>

- Xác thực JWT:

Bước 1: Mỗi khi client gửi yêu cầu (request) đến server, JWT sẽ được gửi kèm theo.

Bước 2: Server nhận JWT và kiểm tra tính hợp lệ của token (kiểm tra chữ ký của token và xem token có hết hạn hay không).

Bước 3: Nếu JWT hợp lệ, server cho phép truy cập vào tài nguyên được yêu cầu; nếu không hợp lệ, server trả về mã lỗi (ví dụ: 401 Unauthorized).

```
const jwt = require('jsonwebtoken');  
  
// Middleware xác thực JWT  
const authenticateJWT = (req, res, next) => {  
  const token = req.header('Authorization')?.split(' ')[1];  
  
  if (!token) {  
    return res.status(403).send('Token is required');  
  }  
  
  jwt.verify(token, 'secretKey', (err, user) => {  
    if (err) {  
      return res.status(403).send('Invalid or expired token');  
    }  
    req.user = user;  
    next();  
  });  
};
```



Hình 2.12 Ví dụ xác thực JWT trong Node.js

jwt.verify(): Hàm này dùng để xác thực JWT. Nó nhận vào token, khóa bí mật và callback để xử lý kết quả.

Nếu token hợp lệ, callback nhận được user (payload).

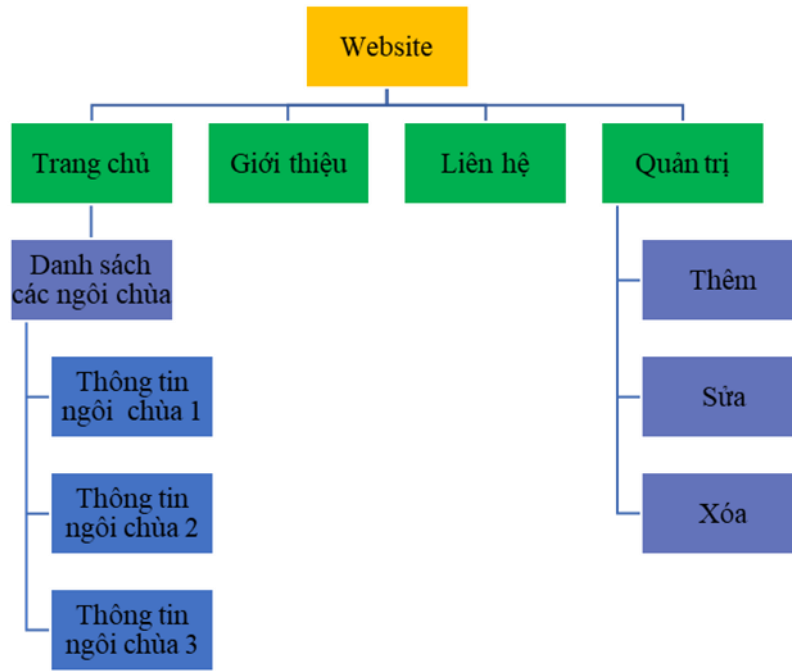
Nếu token không hợp lệ hoặc đã hết hạn, callback nhận được lỗi.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Đặc tả yêu cầu hệ thống

3.1.1. Yêu cầu chức năng

Trang web gồm có các chức năng:



Hình 3.1 Sơ đồ cấu trúc website

Chức năng hiển thị thông tin về tỉnh Trà Vinh: Trang web cần có thông tin chi tiết về tỉnh Trà Vinh, các địa điểm du lịch, văn hóa và lịch sử.

Chức năng giới thiệu các ngôi chùa nổi tiếng tại Trà Vinh: Danh sách các ngôi chùa nổi tiếng, thông tin của mỗi ngôi chùa (gồm tên, mô tả, lịch sử, hình ảnh, vị trí...)

Chức năng quản lý thông tin ngôi chùa: Người quản trị có thể thêm, sửa và xóa thông tin các ngôi chùa (gồm tên, mô tả, lịch sử, hình ảnh, vị trí...).

Chức năng liên hệ: Cho phép người dùng liên hệ để gửi ý kiến về các ngôi chùa, các địa điểm sự kiện...

Tối ưu hóa giao diện: Đảm bảo giao diện thân thiện với người dùng, tối ưu trên tất cả các thiết bị di động và các kích thước màn hình khác nhau.

3.1.2. Yêu cầu phi chức năng

Hiệu suất: Trang web cần tối ưu hóa tốc độ tải trang, giảm thời gian phản hồi.

Tính bảo mật: Đảm bảo an toàn thông tin cho người dùng và hệ thống.

Khả năng mở rộng: Hệ thống cần có khả năng mở rộng trong tương lai để thêm các tính năng mới hoặc mở rộng phạm vi các ngôi chùa.

SEO: Cải thiện khả năng tìm thấy website trên các công cụ tìm kiếm thông qua việc tối ưu hóa nội dung và mã nguồn (meta tags, cấu trúc URL, ...).

3.2. Thiết kế dữ liệu

3.2.1. Mô hình dữ liệu của chùa trong MongoDB

```
13
14 Chua
15 |
16 |— name: "Tên của ngôi chùa (bắt buộc)"
17 |— description: "Mô tả sơ lược về chùa."
18 |— image: "Đường dẫn hình ảnh chùa."
19 |— history:
20 |   |— title: "Tiêu đề lịch sử."
21 |   |— caption:
22 |     |— content: "Nội dung đi kèm với tiêu đề."
23 |     |— img: "Đường dẫn hình ảnh (nếu có) của nội dung."
24 |— googleMapUrl: "Đường dẫn tới bản đồ Google Map."
```

Hình 3.2 Mô hình dữ liệu của chùa trong MongoDB

Chùa (Chua)

- name (String): Tên của ngôi chùa (Bắt buộc).
- description (String): Mô tả về ngôi chùa.
- image (String): Đường dẫn hình ảnh của ngôi chùa.
- history (Array of historySchema): Lịch sử của ngôi chùa, có thể chứa nhiều mục lịch sử.
- googleMapUrl (String): Đường dẫn đến bản đồ Google Map.

Lịch sử (History)

- title (String): Tiêu đề lịch sử.
- caption (Array of captionSchema): Có thể chứa nhiều nội dung của tiêu đề

Chú thích (Caption)

- content (String): Nội dung đi kèm với các tiêu đề.
- img (String): Đường dẫn tới hình ảnh (nếu có) của nội dung

3.2.2. Mô hình dữ liệu người dùng trong MongoDB

```
4
5  users
6  |
7  |— _id: "609c5b6e9e0b8f001c8a56a5"
8  |— username: "Tài khoản"
9  |— password: "mật khẩu"
10 |— role: "vai trò của người dùng (user hoặc admin)"
11
```

Hình 3.3 mô hình dữ liệu người dùng trong MongoDB

username:

- Kiểu dữ liệu: String.
- Yêu cầu: Trường này là bắt buộc và phải có giá trị duy nhất trong cơ sở dữ liệu.

password:

- Kiểu dữ liệu: String.
- Yêu cầu: Trường này là bắt buộc.

role:

- Kiểu dữ liệu: String.
- Giới hạn giá trị: Trường này chỉ chấp nhận một trong hai giá trị là 'user' hoặc 'admin' (sử dụng enum).
- Mặc định: Nếu không chỉ định, mặc định sẽ là 'user'.

3.2.3. Chi tiết các thực thể

Ngôi chùa: Bao gồm các thông tin về tên, mô tả lịch sử, hình ảnh và vị trí. Có thể có nhiều ngôi chùa trong cơ sở dữ liệu, mỗi ngôi chùa có thể có nhiều hình ảnh (nếu cần).

Người quản lý : Quản lý các tài khoản có quyền truy cập và thay đổi thông tin.

3.3. Thiết kế, kiến trúc ứng dụng

3.3.1. Kiến trúc toàn cảnh

Kiến trúc toàn cảnh của hệ thống website giới thiệu tỉnh Trà Vinh và các ngôi chùa nổi tiếng bao gồm ba lớp chính: Frontend, Backend và Database. Frontend được xây dựng bằng React.js để cung cấp giao diện người dùng hiện đại, tối ưu cho thiết bị di động và khả năng tương tác động. Backend sử dụng Node.js và Express để xử lý các yêu cầu từ người dùng và cung cấp API RESTful, bao gồm các phương thức CRUD cho các ngôi chùa, lịch sử và chú thích. Dữ liệu được lưu trữ trong MongoDB, với cấu trúc linh hoạt, giúp dễ dàng lưu trữ các tài liệu liên quan đến ngôi chùa và các thông tin lịch sử. Hệ thống tuân theo kiến trúc 3 lớp, với giao tiếp giữa các thành phần thông qua API, đảm bảo tính hiệu quả, bảo mật và khả năng mở rộng.

3.3.2. Cách frontend nhận dữ liệu

Frontend sẽ nhận dữ liệu từ backend thông qua các API:

GET /chua: Lấy danh sách các ngôi chùa.

GET /chua/:id: Lấy thông tin chi tiết về một ngôi chùa.

POST /chua: Thêm mới một ngôi chùa.

PUT /chua/:id: Cập nhật thông tin một ngôi chùa.

DELETE /chua/:id: Xóa một ngôi chùa.

3.3.3. Thiết kế API

API sẽ thực hiện các chức năng CRUD cho ngôi chùa, người dùng sẽ truy cập các tài nguyên này thông qua các HTTP methods (GET, POST, PUT, DELETE). Các endpoint API sẽ được thiết kế như sau:

/api/chua: Endpoint cho phép lấy danh sách các ngôi chùa (GET), thêm ngôi chùa mới (POST).

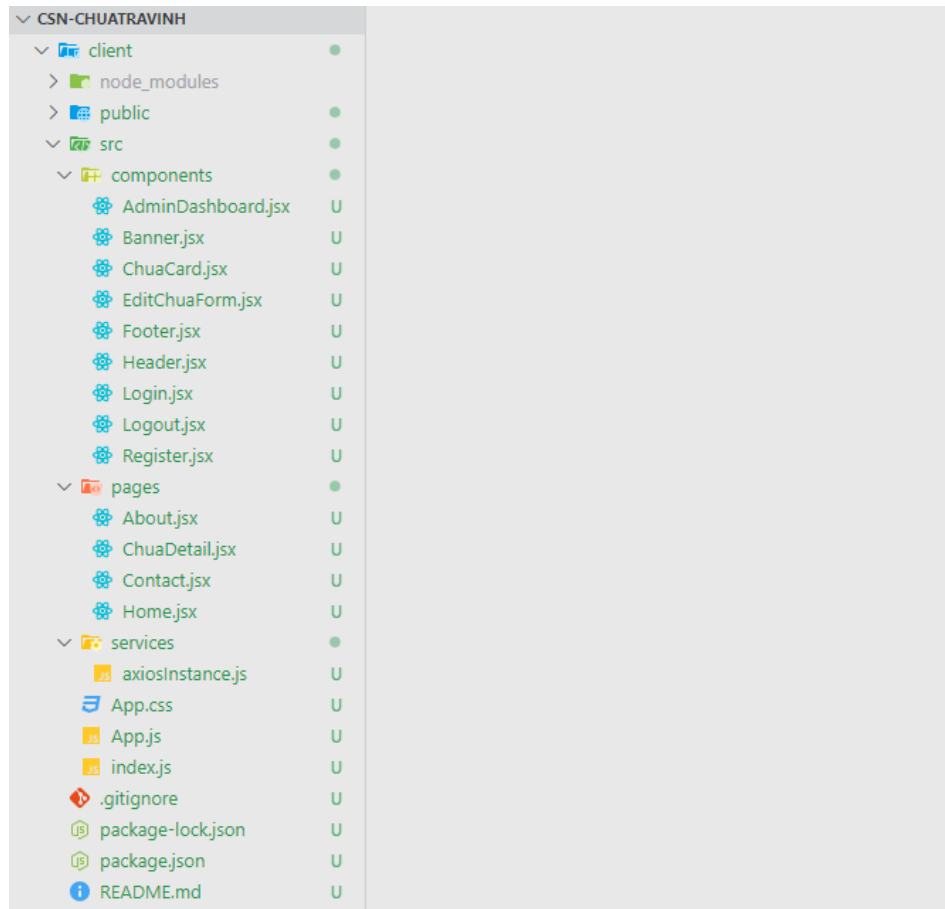
/api/chua/:id: Endpoint cho phép lấy thông tin ngôi chùa chi tiết (GET), sửa thông tin (PUT), hoặc xóa (DELETE).

/api/auth/login : Quản lý người dùng đăng nhập

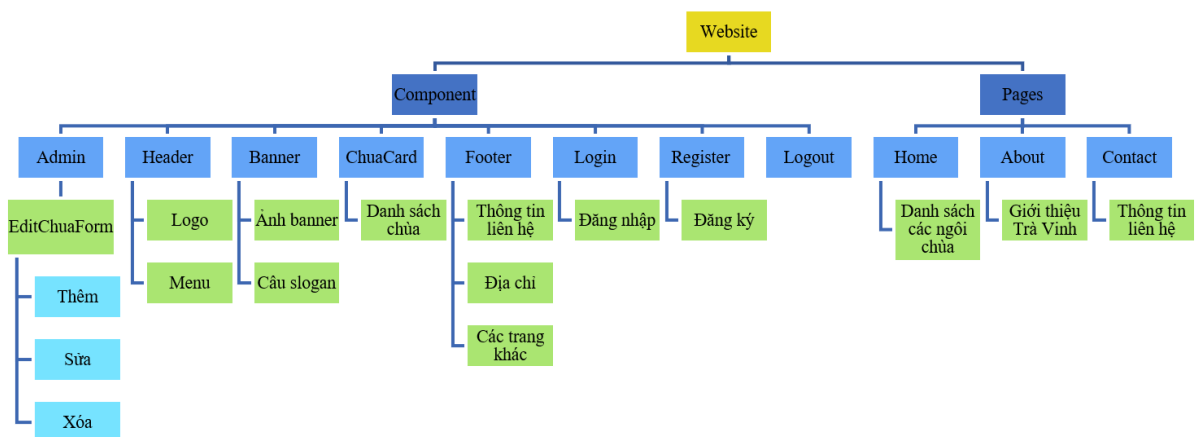
/api/auth/register : Quản lý người dùng đăng ký

3.3.4. Thiết kế frontend

Thiết kế frontend của website giới thiệu tỉnh Trà Vinh và các ngôi chùa nổi tiếng được xây dựng bằng React.js, kết hợp với các công nghệ và phương pháp hiện đại để đảm bảo hiệu suất, tính tương tác và khả năng tối ưu với thiết bị di động.



Hình 3.4 Cấu trúc thư mục frontend



Hình 3.5 Sơ đồ cấu trúc thư mục và chức năng

Các thành phần và trang này sẽ được import vào tập tin App.js

```
client > src > App.js > ...
1  import React from 'react';
2  import { Routes, Route } from 'react-router-dom';
3  import { HelmetProvider } from 'react-helmet-async';
4  import Home from './pages/Home';
5  import Header from './components/Header';
6  import Banner from './components/Banner';
7  import Footer from './components/Footer';
8  import ChuaDetail from './pages/ChuaDetail';
9  import Quantri from './components/AdminDashboard';
10 import Contact from './pages/Contact';
11 import About from './pages/About';
12 import './App.css';
13 import Login from './components/Login';
14 import Register from './components/Register';
15
16 const App = () => {
17   return (
18     <HelmetProvider>
19       <Header />
20       <Banner />
21       <Routes>
22         <Route path="/" element={<Home />} />
23         <Route path="/quantri" element={<Quantri />} />
24         <Route path="/Lienhe" element={<Contact />} />
25         <Route path="/Gioithieu" element={<About />} />
26         <Route path="/dangnhap" element={<Login />} />
27         <Route path="/dangky" element={<Register />} />
28         <Route path="/chua/:id" element={<ChuaDetail />} />
29       </Routes>
30       <Footer />
31     </HelmetProvider>
32   );
33 };
34
35 export default App;
36
```

Hình 3.6 Tập tin App.js

- Chức Năng và Tương Tác:

Frontend sẽ cung cấp các chức năng tương tác động, bao gồm:

Hiển thị dữ liệu từ backend: Sử dụng Axios hoặc Fetch API để kết nối với các API từ backend (Node.js). Dữ liệu sẽ được lấy và hiển thị trên các trang như danh sách ngôi chùa hoặc chi tiết ngôi chùa.

Hiển thị thông tin lịch sử: Mỗi ngôi chùa có thể có nhiều mục lịch sử, được hiển thị dưới dạng danh sách các tiêu đề với mô tả chi tiết kèm theo hình ảnh (nếu có).

Chức năng CRUD cho Admin: Quản trị viên có thể thêm mới, chỉnh sửa và xóa ngôi chùa, lịch sử, chú thích qua giao diện Admin. Các thao tác này sẽ gửi yêu cầu đến API backend để thực hiện thay đổi dữ liệu.

- Giao Diện Người Dùng:

Giao diện hiện đại : xây dựng các thành phần UI như bảng điều khiển, modal, form nhập liệu, để đảm bảo tính thẩm mỹ và thân thiện với người dùng.

Responsive Design: Giao diện sẽ tự động điều chỉnh cho phù hợp với các loại màn hình khác nhau (máy tính, tablet, điện thoại) bằng cách sử dụng CSS Media Queries hoặc CSS Flexbox/Grid.

SEO Tối ưu: Các thành phần như tiêu đề trang, mô tả meta, hình ảnh (với thuộc tính alt) sẽ được tối ưu để giúp website thân thiện với công cụ tìm kiếm (SEO).

Hiệu ứng chuyển động: Sử dụng CSS animations hoặc thư viện như Framer Motion để tạo hiệu ứng chuyển động mượt mà cho các thành phần khi người dùng tương tác với chúng (ví dụ: hover effect, fade-in khi tải trang).

- Tối ưu hiệu suất:

Lazy Loading: Tải các thành phần hoặc dữ liệu chỉ khi người dùng yêu cầu (ví dụ: hình ảnh hoặc danh sách các ngôi chùa).

Code Splitting: Chia nhỏ mã nguồn của ứng dụng để giảm thiểu thời gian tải trang, sử dụng React.lazy và React Suspense.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Kết quả đạt được

Sau quá trình thực hiện đồ án, hệ thống website giới thiệu về tỉnh Trà Vinh và các ngôi chùa nổi tiếng đã được xây dựng và đạt được các kết quả sau:

4.1.1. Hoàn thiện chức năng chính:

Trang chủ hiển thị thông tin tổng quan về tỉnh Trà Vinh với giao diện hiện đại, thân thiện.

Chức năng thêm, sửa, xóa thông tin các ngôi chùa được quản lý thông qua giao diện Admin Dashboard.

Hiển thị thông tin chi tiết về từng ngôi chùa, bao gồm mô tả, hình ảnh, lịch sử và đường dẫn Google Map.

Tích hợp chức năng đăng ký, đăng nhập và phân quyền người dùng (user, admin).

Hiển thị thông tin liên hệ và trang "Giới thiệu" về Trà Vinh.

4.1.2. Tối ưu hóa hiệu suất:

Sử dụng React.js kết hợp với Node.js và MongoDB để đảm bảo tốc độ tải nhanh, trải nghiệm mượt mà.

Tối ưu hóa giao diện với CSS và tương thích với các thiết bị di động, đáp ứng tốt các tiêu chuẩn thiết kế hiện đại.

Tích hợp SEO cơ bản giúp nâng cao khả năng xuất hiện trên công cụ tìm kiếm.

4.1.3. Bảo mật và quản lý dữ liệu:

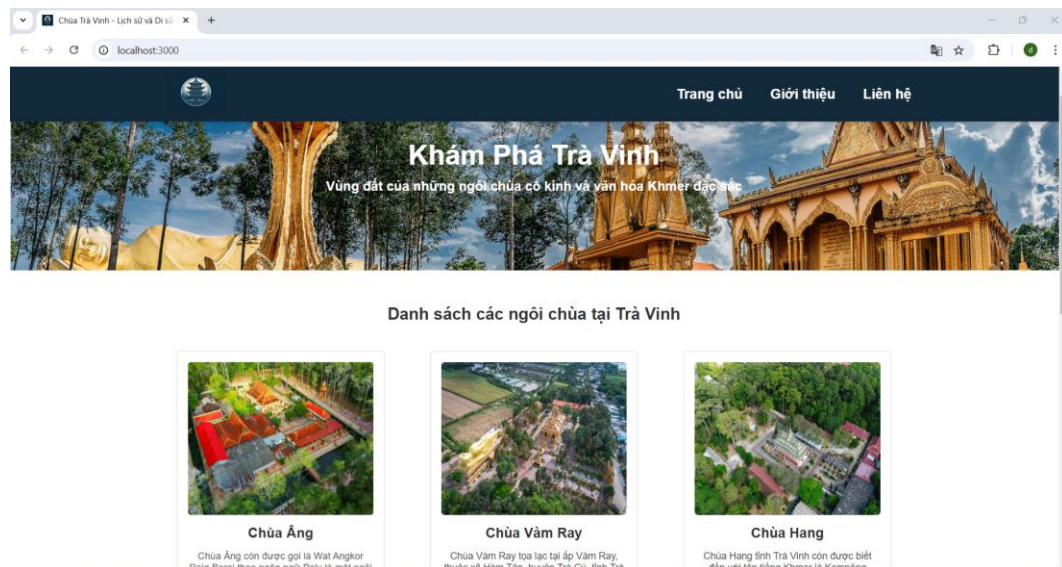
Thông tin người dùng được bảo mật bằng cách mã hóa mật khẩu.

Cấu trúc cơ sở dữ liệu với MongoDB đảm bảo lưu trữ dữ liệu linh hoạt và hiệu quả.

4.2. Giao diện chức năng

Trang chủ (Home Page):

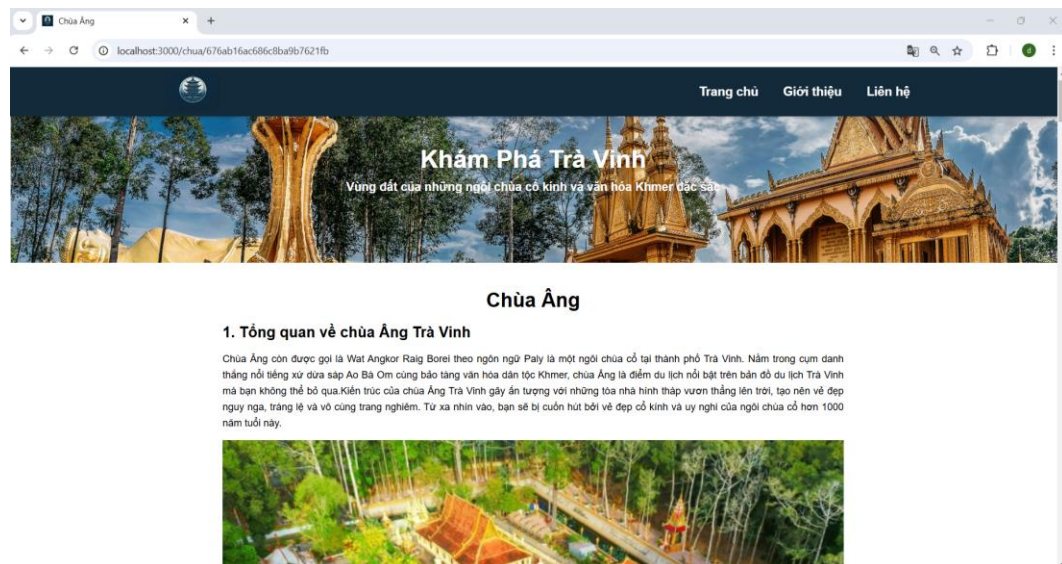
Hiển thị banner quảng bá, thông tin nổi bật về tỉnh Trà Vinh và danh sách các ngôi chùa nổi tiếng.



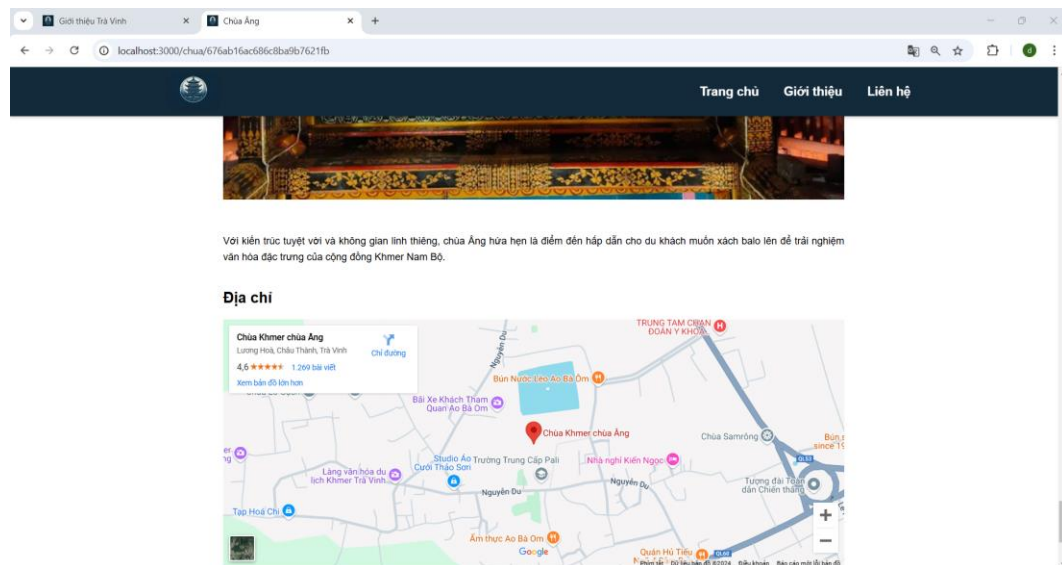
Hình 4.1 Trang chủ

Chi tiết ngôi chùa:

Giao diện hiển thị đầy đủ thông tin về tên, mô tả, hình ảnh, lịch sử chi tiết và bản đồ.



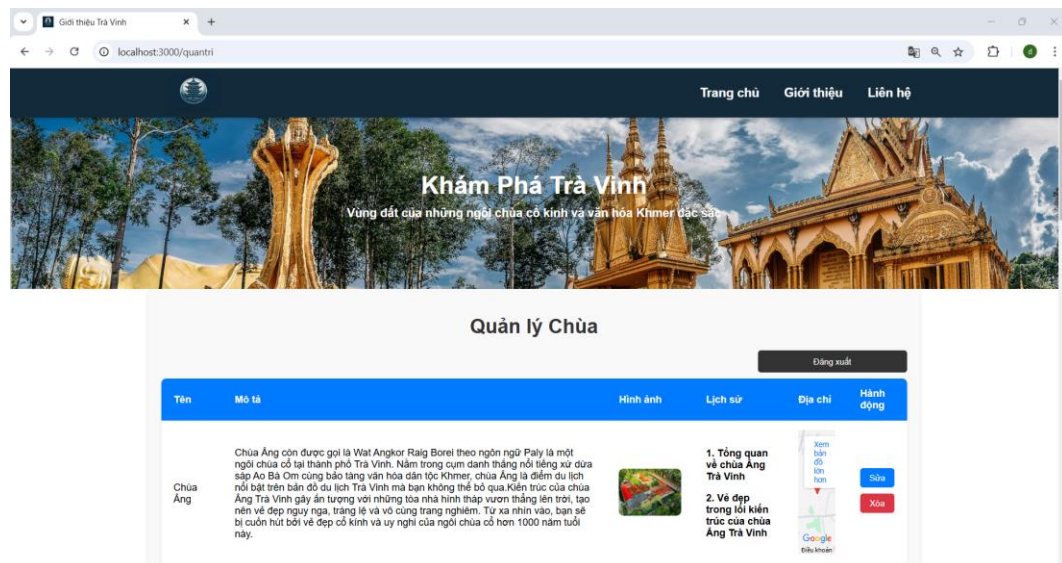
Hình 4.2 Trang thông tin chi tiết về chùa



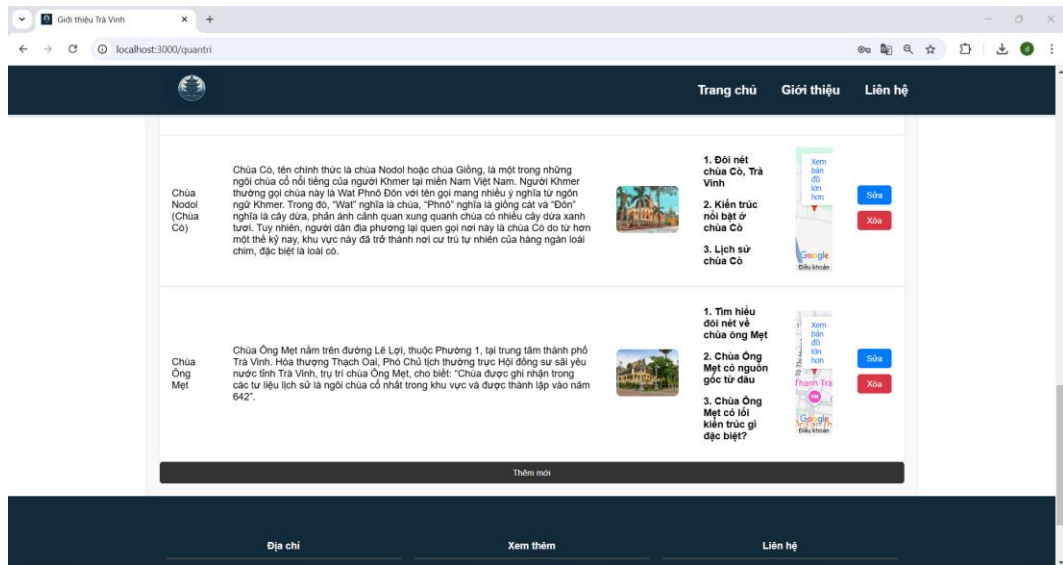
Hình 4.3 Trang thông tin chi tiết bản đồ chùa

Quản lý ngôi chùa (Admin Dashboard):

Chức năng thêm mới, chỉnh sửa, xóa các thông tin về ngôi chùa.

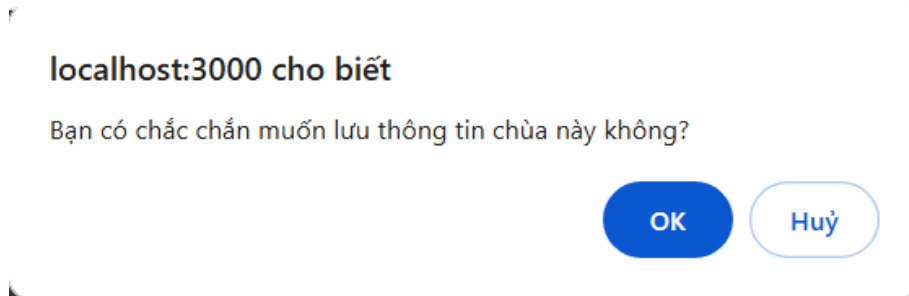


Hình 4.4 Trang quản lý chùa

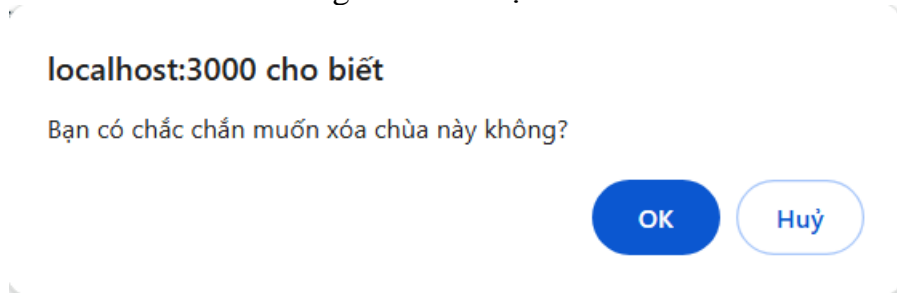


Hình 4.5 Trang quản lý chùa

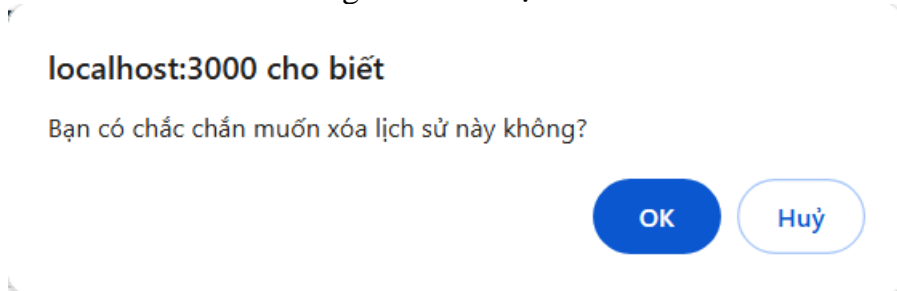
Tích hợp các thông báo xác nhận khi thực hiện thao tác.



Hình 4.6 Thông báo xác nhận khi thêm chùa



Hình 4.7 Thông báo xác nhận khi xóa chùa

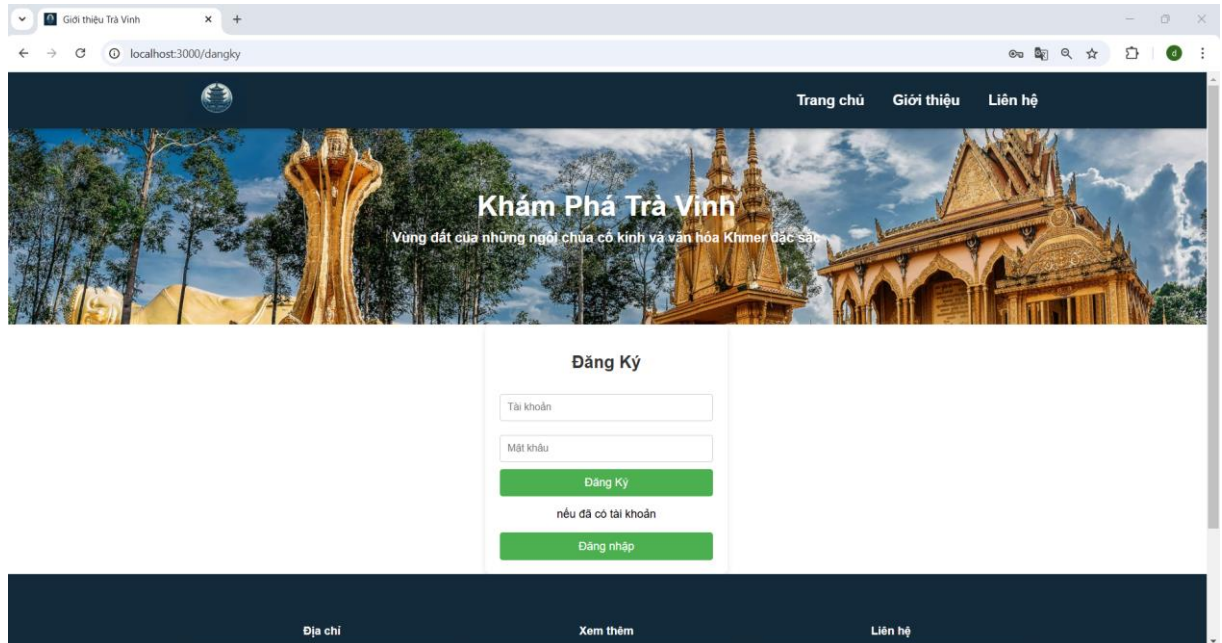


Hình 4.8 Thông báo xác nhận khi xóa 1 lịch sử chùa

Đăng ký/Đăng nhập:

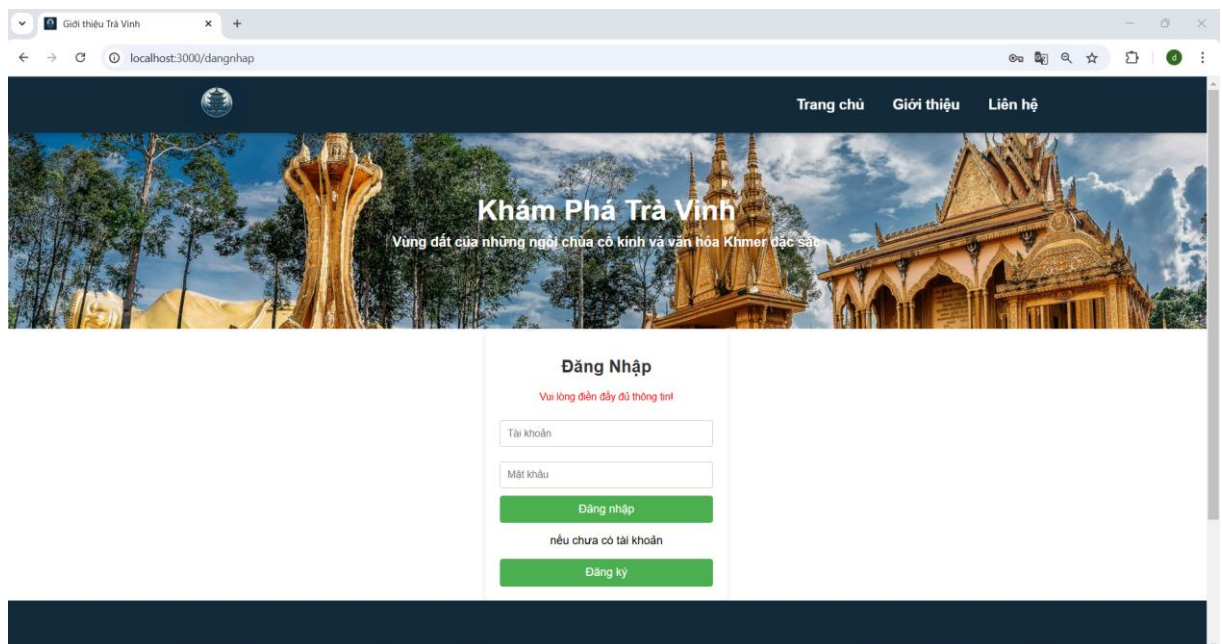
Giao diện thân thiện với chức năng xác thực và thông báo lỗi nếu thông tin không hợp lệ.

Giao diện đăng ký:



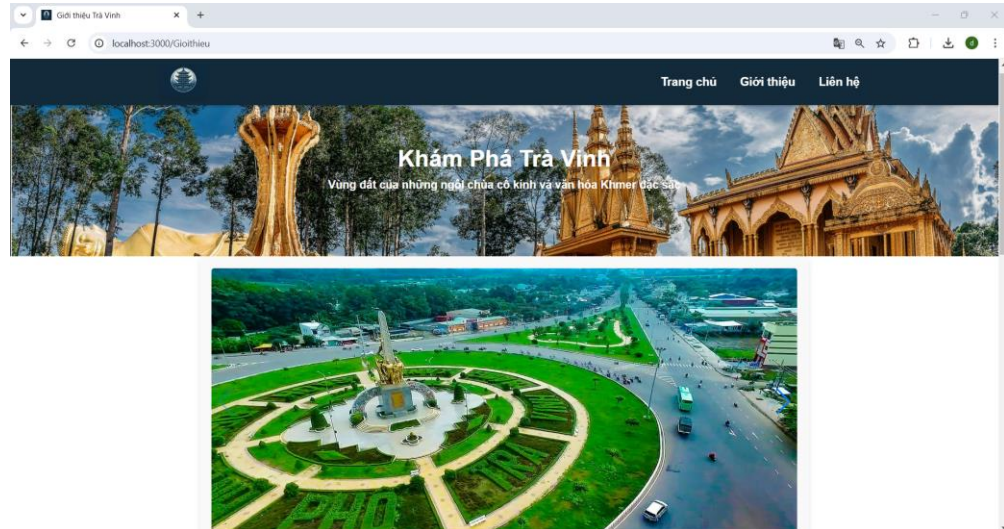
Hình 4.9 Giao diện đăng ký

Giao diện đăng nhập (sẽ thông báo nếu thông tin không có hoặc không hợp lệ):

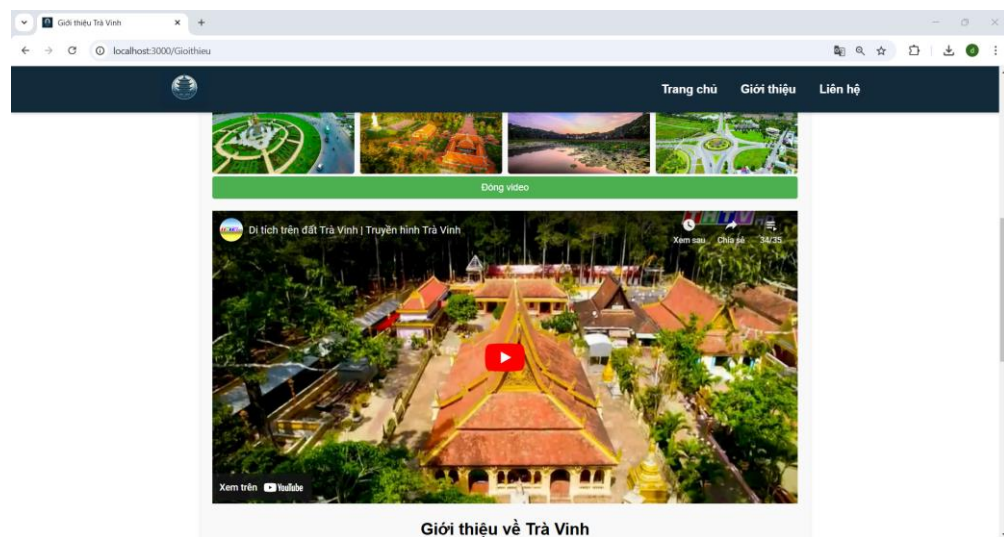


Hình 4.10 Giao diện đăng nhập

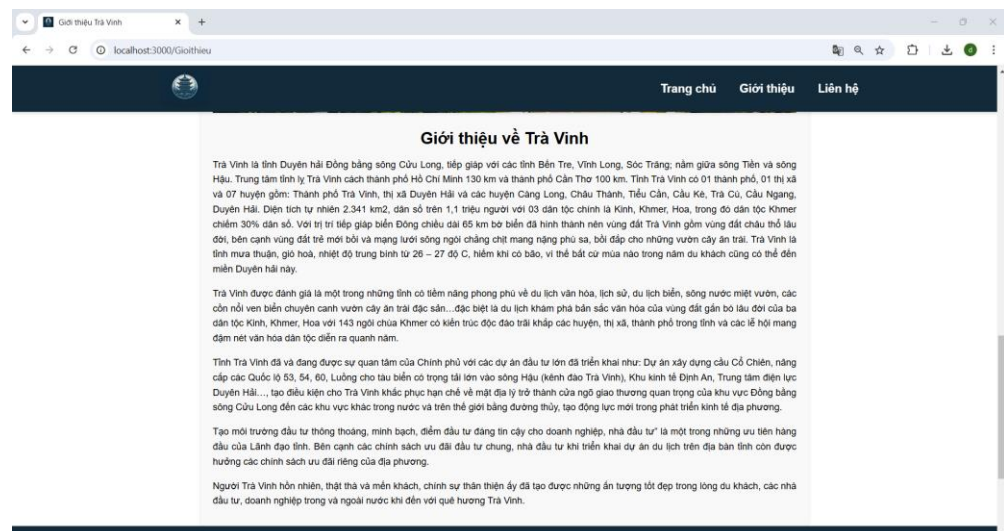
Trang giới thiệu (About):



Hình 4.11 Giao diện trang giới thiệu



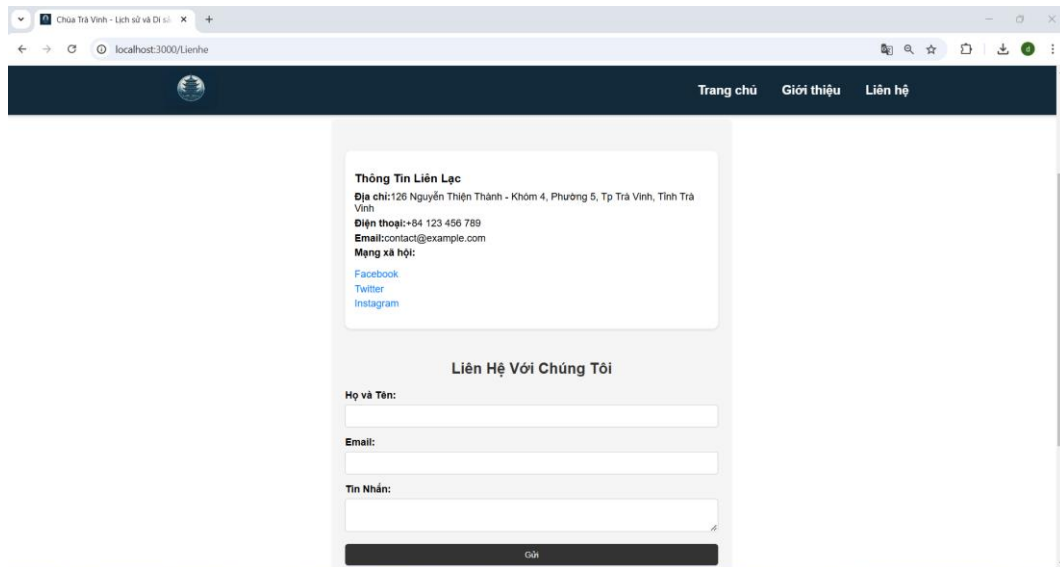
Hình 4.12 Giao diện trang giới thiệu



Hình 4.13 Giao diện trang giới thiệu

Liên hệ (Contact):

Cung cấp biểu mẫu liên hệ và thông tin liên lạc.



Hình 4.14 Giao diện trang liên hệ

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Những kết quả đạt được

Quá trình thực hiện đề tài Xây dựng website giới thiệu về tỉnh Trà Vinh và các ngôi chùa nổi tiếng đã giúp tôi củng cố và mở rộng kiến thức trong lĩnh vực lập trình web. Tôi đã nắm vững các ngôn ngữ frontend như HTML, CSS, JavaScript, và sử dụng React.js để xây dựng các component động, tối ưu hóa hiệu suất và tương tác của website. Việc sử dụng Node.js và MongoDB giúp tôi xây dựng hệ thống quản lý nội dung và lưu trữ dữ liệu hiệu quả. Quá trình tối ưu hóa SEO đã cải thiện khả năng tìm kiếm của website.

Về phần công nghệ :

ReactJS: Sử dụng ReactJS để xây dựng giao diện người dùng một cách linh hoạt và hiệu quả. ReactJS giúp tôi tạo ra các component tái sử dụng, tối ưu hóa quá trình phát triển và quản lý trạng thái của ứng dụng, đồng thời cải thiện hiệu suất và trải nghiệm người dùng với khả năng cập nhật giao diện động.

Express và Node.js: Hiểu rõ về Express và Node.js để xây dựng và xử lý các yêu cầu HTTP, tương tác với dữ liệu MongoDB và xây dựng API. Node.js cung cấp môi trường server-side JavaScript mạnh mẽ, trong khi Express giúp đơn giản hóa việc xử lý các route và quản lý các request-response hiệu quả.

MongoDB: Sử dụng MongoDB để lưu trữ và truy xuất dữ liệu về các ngôi chùa, lịch sử và các thông tin khác. MongoDB với mô hình dữ liệu NoSQL đã giúp tôi dễ dàng quản lý và mở rộng cơ sở dữ liệu mà không gặp khó khăn trong việc thay đổi cấu trúc dữ liệu.

CSS & HTML: Áp dụng các kỹ thuật CSS hiện đại để tạo giao diện đẹp mắt, dễ sử dụng và tương thích với nhiều thiết bị. HTML được sử dụng để xây dựng cấu trúc của trang web, đồng thời đảm bảo website đáp ứng các tiêu chuẩn web và tối ưu trải nghiệm người dùng.

Tối ưu SEO: Áp dụng các kỹ thuật tối ưu SEO cơ bản như sử dụng thẻ meta, tiêu đề trang hợp lý và cấu trúc URL thân thiện với SEO, giúp nâng cao khả năng tìm kiếm và tối ưu hóa website trên các công cụ tìm kiếm.

Về phần trang web :

Trang web được xây dựng nhằm giới thiệu về tỉnh Trà Vinh và các ngôi chùa nổi tiếng tại đây, bao gồm lịch sử, thông tin các ngôi chùa. Các chức năng chính của trang web bao gồm:

Giới thiệu lịch sử và thông tin các ngôi chùa: Cung cấp thông tin chi tiết về từng ngôi chùa, bao gồm lịch sử, kiến trúc, văn hóa, vị trí.

Quản lý nội dung: Hệ thống quản trị nội dung cho phép thêm, sửa, xóa thông tin về các ngôi chùa một cách dễ dàng.

Cấu trúc và giao diện:

Giao diện người dùng (UI) được thiết kế trực quan, dễ sử dụng và thân thiện với người dùng. Các trang chính bao gồm: trang chủ, trang giới thiệu, trang thông tin các ngôi chùa, và trang liên hệ.

Responsive design: Giao diện website được tối ưu hóa để hiển thị tốt trên các thiết bị khác nhau, bao gồm máy tính để bàn, laptop, và thiết bị di động.

Sử dụng CSS và HTML: HTML được sử dụng để xây dựng cấu trúc trang web, còn CSS3 giúp tạo kiểu dáng bắt mắt và chuyên nghiệp. Các hiệu ứng chuyển động và thiết kế hiện đại giúp website trở nên sinh động và dễ tiếp cận.

5.2. Hướng phát triển

Mặc dù đã tạo ra một trang web với nhiều chức năng, tuy nhiên vì thời gian có hạn, trang web vẫn còn một số hạn chế chưa đáp ứng được tất cả yêu cầu của người dùng, trong tương lai tôi sẽ tiếp tục phát triển và cải thiện trang web, khắc phục các lỗi nhỏ và bổ sung các tính năng mới còn thiếu sót trong trang web cụ thể:

- Thêm chức năng tìm kiếm và lọc thông tin chưa hoàn thiện.
- Tích hợp đầy đủ các sự kiện và thông báo.
- Thêm tính năng đa ngôn ngữ.
- Bổ sung nhiều kho tài liệu và hình ảnh đa dạng.
- Thêm tính năng bình luận và đánh giá các ngôi chùa.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] <https://chat.openai.com/>
- [2] <https://jwt.io/introduction>
- [3] <https://aws.amazon.com/vi/what-is/restful-api/>
- [4] <https://www.w3schools.com/react/>
- [5] <https://200lab.io/blog/reactjs-la-gi/>
- [6] <https://aws.amazon.com/vi/what-is/api>