BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH : CÔNG NGHỆ THÔNG TIN

Sinh viên : ĐỖ XUÂN TOÀN

GVHD : TS. LƯƠNG THANH NHẠN

HẢI PHÒNG – 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

XÂY DỰNG ỨNG DỤNG NỀN TẢNG WEB TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN BẢN ĐỒ

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : ĐỖ XUÂN TOÀN

GVHD : TS. LƯƠNG THANH NHẠN

HẢI PHÒNG – 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: **Đỗ xuân Toàn** **Mã SV**: 2012111015

Lớp : CT2401C

Ngành : Công nghệ thông tin

Tên đề tài: Xây dựng ứng dụng nền tảng web tìm đường đi ngắn nhất trên bản đồ

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a. Mô tả tóm tắt đề tài

Tìm hiểu về bài toán, phân tích thiết kế hệ thống và xây dựng website tìm đường đi ngắn nhất trên bản đồ.

b. Nội dung hướng dẫn

- Tìm hiểu về hiện trạng và bài toán tìm đường đi ngắn nhất trên bản đồ.
- Phân tích, thiết kế cơ sở dữ liệu, hệ thống
- Cài đặt ứng dụng và thử nghiệm.
- Nhận xét, đánh giá và kết luận

c. Kết quả cần đạt được

- Tài liệu mô tả các kết quả đã thực hiện
- Website tìm đường đi ngắn nhất

2. Các tài liệu, số liệu cần thiết

- Tài liệu tham khảo về hệ quản trị cơ sở dữ liệu, ngôn ngữ lập trình web
- Tài liệu tham khảo về phân tích và thiết kế hệ thống thông tin
- Các trang web hiện có về đường đi ngắn nhất trên bản đồ.

3. Địa điểm thực tập tốt nghiệp

-

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Họ và tên : Lương Thanh Nhạn

Học hàm, học vị : Tiến sĩ

Cơ quan công tác : Trường Đại học Y Dược Hải Phòng

Nội dung hướng dẫn:

- Tìm hiểu về bài toán
- Thực hiện phân tích thiết kế hệ thống
- Tìm hiểu một hệ quản trị cơ sở dữ liệu, một ngôn ngữ lập trình web để xây dựng hệ thống.
- Cài đặt ứng dụng và thử nghiệm.
- Nhận xét, đánh giá và kết luận

Kết quả cần đạt được

- Tài liệu mô tả các kết quả đã thực hiện
- Website đường đi ngắn nhất trên bản đồ

Đề tài tốt nghiệp được giao ngày tháng năm 2024

Yêu cầu phải hoàn thành xong trước ngày tháng năm 2024

Đã nhận nhiệm vụ ĐTTN	**Đã giao nhiệm vụ ĐTTN**
Sinh viên	Giảng viên hướng dẫn

TS. Lương Thanh Nhạn

Hải Phòng, ngày tháng năm 2024

TRƯỞNG KHOA

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

Họ và tên giảng viên: Lương Thanh Nhạn

Đơn vị công tác: Trường Đại học Y Dược Hải Phòng.

Họ và tên sinh viên: Đỗ Xuân Toàn

Ngành: Công nghệ Thông tin

Nội dung hướng dẫn:

- Tìm hiểu về bài toán
- Thực hiện phân tích thiết kế hệ thống
- Tìm hiểu hệ quản trị cơ sở dữ liệu, ngôn ngữ lập trình web để xây dựng website tìm đường đi ngắn nhất trên bản đồ.
- Nhận xét, đánh giá và kết luận

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp

- Sinh viên có ý thức tốt, có tinh thần cố gắng trong quá trình làm đồ án tốt nghiệp. Từ việc sưu tập, tìm hiểu, tổng hợp tài liệu và tìm hiểu bài toán, sinh viên đã vận dụng các kiến thức đã học để phân tích thiết kế hệ thống và xây dựng website tìm đường đi ngắn nhất trên bản đồ.
- Trong quá trình thực hiện đồ án tốt nghiệp, sinh viên luôn cố gắng để đảm bảo đúng tiến độ thực hiện theo quy định của Nhà trường và hướng dẫn của giáo viên hướng dẫn.

2. Đánh giá chất lượng của đồ án/khóa luận (so với nội dung yêu cầu đó đề ra trong nhiệm vụ Đ.T.T.N trên các mặt lý luận, thực tiễn, tính toán số liệu…)

- Đồ án tốt nghiệp của sinh viên đã đáp ứng được các yêu cầu của đề cương đồ án tốt nghiệp đã đặt ra.
- Phần lý thuyết đã cơ bản đáp ứng được yêu cầu tổng quan kiến thức chung và tìm hiểu chi tiết về bài toán cần giải quyết.
- Phần chương trình thử nghiệm đã phần nào thể hiện được khả năng vận dụng những kiến thức đã được học, tìm hiểu vào giải quyết bài toán thực tế.

3. Ý kiến của giảng viên hướng dẫn tốt nghiệp

Đạt [X] Không đạt [ ] Điểm: ……...

Hải Phòng, ngày tháng năm 2024.

Giảng viên hướng dẫn

TS. Lương Thanh Nhạn

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

Họ và tên giảng viên: .....

Đơn vị công tác: .....

Họ và tên sinh viên: Ngành: Công nghệ thông tin

Đề tài tốt nghiệp:

1. Phần nhận xét của giảng viên chấm phản biện

……

……

……

……

……

2. Những mặt còn hạn chế

……

……

……

……

3. Ý kiến của giảng viên chấm phản biện

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm:…….

Hải Phòng, ngày.....tháng …… năm 2024

Giảng viên chấm phản biện

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành đồ án này trước tiên em xin chân thành cảm ơn đến các quý thầy, cô giáo trường Đại Học Và Quản Lý Công Nghệ Hải Phòng đặc biệt là các thầy, cô khoa Công nghệ thông tin đã tạo điều kiện và cơ hội cho em thực hiện đồ án tốt nghiệp trong lĩnh vực Phần mềm Công nghệ Thông tin. Đặc biệt, em xin gửi đến cô TS. Lương Thanh Nhạn người đã tận tình hướng dẫn, giúp đỡ em trong suốt quá trình làm đồ án lời cảm ơn sâu sắc nhất.

Mặc dù đã rất cố gắng trong việc nghiên cứu và thực hiện đồ án một cách hoàn chỉnh nhất, nhưng do thời gian và kiến thức của em còn hạn chế nên đồ án không thể tránh khỏi nhiều thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của thầy cô để em có thể hoàn thiện đồ án này tốt hơn.

Em xin chân thành cảm ơn!

Hải Phòng, ngày …… tháng …… năm 2024.

Sinh viên

LỜI CAM ĐOAN

Em xin cam đoan đề tài "Xây dựng ứng dụng nền tảng web tìm đường đi ngắn nhất trên bản đồ" trong đồ án tốt nghiệp của em được tiến hành một cách công khai và minh bạch, dựa trên sự cố gắng và nỗ lực của bản thân cũng như sự giúp đỡ, hướng dẫn tận tình của giáo viên hướng dẫn TS. Lương Thanh Nhạn. Các số liệu nghiên cứu nêu trong đồ án đảm bảo tính trung thực, không sao chép hay sử dụng kết quả của bất kỳ công trình nào đã được công bố trước đây. Nếu phát hiện có sự sao chép, em xin chịu hoàn toàn trách nhiệm và kỷ luật từ phía nhà trường.

Hải Phòng, ngày …… tháng …… năm 2024.

Sinh viên

MỤC LỤC

DANH MỤC VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Cụm từ đầy đủ
1	HTML	HyperText Markup Language
2	CSS	Cascading Style Sheets
3	API	Application Programming Interface
4	I/O	Input/ Output
5	HTTP	HyperText Transfer Protocol
6	JSON	JavaScript Object Notation
7	OSM	OpenStreetMap
8	id	Identification
9	MB	Megabyte
10	OS	Operating System
11	CPU	Central Processing Unit
12	RAM	Random Access Memory
13	GPS	Global Positioning System
14	WIFI	Wireless Fidelity

DANH SÁCH HÌNH VẼ

DANH MỤC BẢNG

MỞ ĐẦU

1. Lý do chọn đề tài

Trong thời đại hiện nay, công nghệ đã trở thành một phần không thể thiếu trong mọi lĩnh vực của cuộc sống. Việc sử dụng công nghệ thông tin đã và đang thúc đẩy sự phát triển và cải thiện hiệu suất làm việc trong nhiều lĩnh vực. Trong lĩnh vực tìm đường, việc áp dụng công nghệ thông tin có thể đem lại nhiều lợi ích và giải pháp cho các thách thức hiện tại.

Trong quá khứ, việc tìm đường đi giữa hai địa điểm đã gặp phải nhiều hạn chế và thách thức. Trước đây, việc tìm đường thường phụ thuộc vào bản đồ giấy, hướng dẫn từ con người, hoặc sử dụng các hệ thống điều hướng cố định có hạn chế về dữ liệu hoặc tính năng. Công nghệ thông tin đã đóng vai trò quan trọng trong việc giải quyết những hạn chế này và cải thiện quá trình tìm đường đi.

Sự phát triển của công nghệ thông tin đã mở ra những cơ hội mới trong việc tìm đường đi. Các dịch vụ bản đồ trực tuyến và ứng dụng điều hướng như Google Maps, OpenStreetMap đã cung cấp dữ liệu chính xác và cập nhật, cho phép người dùng tìm kiếm và lập kế hoạch đường đi một cách nhanh chóng và thuận tiện hơn.

Chính vì lẽ đó, em đã chọn đề tài "Tìm đường đi ngắn nhất trên bản đồ" cho dự án của mình. Mục tiêu của dự án là nghiên cứu và triển khai các thuật toán tìm đường đi như Dijkstra và tích hợp chúng vào một ứng dụng web. Em tin rằng việc áp dụng công nghệ thông tin trong lĩnh vực này sẽ cung cấp một giải pháp hiệu quả và tiện ích cho người dùng, giúp họ dễ dàng tìm ra đường đi ngắn nhất giữa hai địa điểm một cách nhanh chóng và hiệu quả.

2. Nội dung nghiên cứu

- Nghiên cứu và phân tích các website tìm đường đi.
- Xác định yêu cầu và đề xuất xây dựng website.
- Tìm hiểu về dữ liệu OpenStreetMap và cách sử dụng trong dự án.
- Tìm hiểu cách hiển thị bản trên website qua thư viện Leaflet.
- Tìm hiểu thuật toán tìm đường đi ngắn nhất.
- Lập trình website.

3. Mục đích chọn đề tài

Mục tiêu chính của đề tài là nghiên cứu và phát triển một website nền tảng ứng dụng tìm đường đi ngắn nhất trên bản đồ. Mục đích của dự án là cung cấp cho người dùng một công cụ hiệu quả và tiện lợi để tìm kiếm địa điểm, lập kế hoạch đường đi và hiển thị tuyến đường ngắn nhất giữa hai điểm trên bản đồ. Đồng thời, thông qua việc nghiên cứu và triển khai, em mong muốn nâng cao sự tiện ích và khả năng áp dụng của công nghệ thông tin trong việc giải quyết vấn đề di chuyển và giao thông trong cuộc sống hàng ngày của mọi người.

4. Phương pháp nghiên cứu

a) Phương pháp tiếp cận

Tìm hiểu và phân tích các tài liệu, sách vở, bài báo và tài liệu trực tuyến liên quan đến các thuật toán và phương pháp tìm đường đi ngắn nhất trên bản đồ.

b) Phương pháp thực nghiệm

Thực hiện các thử nghiệm và thí nghiệm trên website mẫu (google maps, bing maps) để đánh giá hiệu suất và tính năng của hệ thống tìm đường đi.

c) Phương pháp phân tích

Phân tích các dữ liệu thu thập được từ các thử nghiệm và thí nghiệm để đánh giá hiệu suất và tính chính xác của hệ thống tìm đường đi.

d) Nghiên cứu phát triển

Nghiên cứu và phát triển các thuật toán và phương pháp mới để cải thiện hiệu suất và tính ứng dụng của hệ thống tìm đường đi.

e) Tư duy thiết kế

Áp dụng tư duy thiết kế để phát triển các giải pháp mới và sáng tạo trong việc xây dựng website tìm đường đi.

5. Phạm vi, đối tượng nghiên cứu của đề tài

a) Phạm vi nghiên cứu

Phát triển một website hoàn chỉnh tìm đường đi ngắn nhất trên bản đồ.

b) Đối tượng sử dụng

Bao gồm tất cả mọi người (những người dùng thông thường và các nhóm đặc biệt như những người di chuyển hàng ngày, du khách, nhân viên giao hàng, v.v).

c) Phạm vi nghiên cứu

Địa lý: Khu vực nhỏ thành phố Hải phòng - Việt Nam.

Công nghệ: JavaScript, HTML, CSS, Nodejs, dữ liệu bản đồ của OpenStreetMap.

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Hiện trạng hệ thống tìm đường đi ngắn nhất

Trong thời đại công nghệ ngày càng phát triển, các dịch vụ tìm đường trực tuyến như Google Maps, Bing Maps và OpenStreetMap đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta. Các dịch vụ này mang lại nhiều ưu điểm, bao gồm giao diện trực quan, tính năng tìm kiếm mạnh mẽ và việc cập nhật dữ liệu nhanh chóng, từ đó giúp người dùng dễ dàng lập kế hoạch và thực hiện các chuyến đi của mình.

Tuy nhiên, không phải lúc nào các dịch vụ này cũng đáp ứng được mọi nhu cầu của người dùng. Mặc dù các trải nghiệm người dùng trên Google Maps [6], Bing Maps [7] và OpenStreetMap [8] mang lại trải nghiệm tuyệt vời cho người dùng, nhưng vẫn tồn tại những hạn chế đáng lưu ý. Một trong những thách thức phổ biến là việc người dùng cần thay đổi địa điểm bắt đầu hoặc địa điểm đến khi tìm kiếm đường đi. Thay vì có thể chọn trực tiếp trên bản đồ, người dùng thường phải nhấp và di chuyển biểu tượng vị trí của điểm bắt đầu hoặc điểm đến đến vị trí mong muốn. Điều này không chỉ có thể trở nên phức tạp và tốn thời gian đặc biệt là trên thiết bị di động, mà còn gây khó khăn cho những người không quen thuộc với công nghệ hoặc có khả năng thao tác hạn chế.

Với nhận thức về những khó khăn này, đề tài của em tập trung vào việc nghiên cứu và đề xuất xây dựng một ứng dụng trên nền tảng web tìm đường đi ngắn nhất. Ứng dụng của em sẽ tích hợp trải nghiệm gợi ý tìm kiếm thông minh và cho phép người dùng dễ dàng thay đổi địa điểm bắt đầu hoặc địa điểm đến bằng cách chạm trực tiếp trên bản đồ. Điều này giúp người dùng có thể chọn và tìm đường đi một cách thuận tiện hơn, giải quyết những thách thức thực tế trong việc di chuyển hàng ngày.

1.2. Phát biểu bài toán

Trong thời đại công nghệ số hiện nay, việc tìm đường đi ngắn nhất giữa hai điểm trên bản đồ là một yêu cầu phổ biến và cần thiết. Đặc biệt, với sự phát triển của các ứng dụng di động và trang web, việc cung cấp một giao diện đơn giản và hiệu quả để tìm đường đi đã trở thành một yêu cầu ngày càng tăng.

Bài toán được xây dựng nhằm mục đích cung cấp cho người dùng một công cụ thuận tiện và nhanh chóng để tìm đường đi từ một điểm đến một điểm khác. Với ứng dụng này, người dùng có thể dễ dàng xác định con đường tối ưu dựa trên loại phương tiện di chuyển, điểm xuất phát và điểm đến.

Do vậy bài toán cần đạt được các yêu cầu sau:

Người dùng tìm đường đi ngắn nhất giữa hai điểm trên bản đồ theo yêu cầu cụ thể như: chọn phương tiện đi lại (ô tô, xe máy, đi bộ), chọn điểm bắt đầu, và điểm đến. Người dùng có thể xem thông tin tuyến đường đi ngắn nhất giữa hai điểm trên bản đồ theo phương tiện đã chọn.

1.3. Giải pháp

Tích hợp nhiều dịch vụ địa điểm: Phát triển hệ thống có khả năng tích hợp nhiều dịch vụ địa điểm khác nhau như quán cafe, nhà hàng, bệnh viện, trạm xăng, v.v. để người dùng có thể dễ dàng tìm kiếm và lập kế hoạch cho các chuyến đi của mình.

Cung cấp tùy chọn đa dạng: Tạo ra giao diện người dùng linh hoạt và dễ sử dụng, cho phép người dùng lựa chọn từ nhiều địa điểm và yêu cầu tìm kiếm địa điểm theo nhu cầu cụ thể của họ.

Hiển thị đường đi chi tiết: Tạo ra các tùy chọn hiển thị đường đi chi tiết và tối ưu cho người đi bộ, xe máy hoặc ô tô.

1.4. Yêu cầu đạt được của hệ thống

❖ **Yêu cầu chung**

- ✓ Tích hợp đa dịch vụ: Hệ thống cần tích hợp nhiều dịch vụ địa điểm khác nhau để cung cấp cho người dùng một trải nghiệm toàn diện khi tìm kiếm địa điểm.
- ✓ Giao diện thân thiện: Giao diện người dùng cần được thiết kế dễ sử dụng và trực quan, phản ánh yêu cầu của người dùng và cung cấp trải nghiệm tìm kiếm thuận tiện.

❖ **Yêu cầu của các chức năng**

- ✓ Tùy chọn phương tiện đi: Cung cấp tùy chọn cho người dùng để lựa chọn loại đường đi phù hợp như đi bộ, xe máy hoặc ô tô.
- ✓ Chọn điểm bắt đầu, điểm đến: Cho phép người dùng chọn địa điểm bắt đầu và địa điểm đến trên bản đồ hoặc thông qua việc nhập liệu.

- ✓ Tìm kiếm thông minh: Cung cấp gợi ý tìm kiếm thông minh dựa trên các từ khóa mà người dùng nhập vào, giúp họ dễ dàng tìm kiếm địa điểm bắt đầu và địa điểm đến.
- ✓ Chọn điểm trên bản đồ: Cho phép người dùng chạm trực tiếp trên bản đồ để chọn điểm bắt đầu hoặc điểm đến mong muốn, tăng tính tiện lợi và trực quan cho trải nghiệm người dùng.
- ✓ Hiển thị thông tin địa bắt đầu và điểm đến: Cung cấp thông tin về tên của địa điểm hiển thị, biểu tượng đánh dấu địa chỉ các điểm đó trên bản đồ.
- ✓ Hiển thị đường đi: Tạo và hiển thị đường đi ngắn nhất.

1.5. Giới thiệu các phần mềm sử dụng để cài đặt chương trình

❖ Visual Studio Code

Miễn phí và mã nguồn mở: Visual Studio Code là một trình soạn thảo mã nguồn mở và miễn phí, có sẵn cho Windows, macOS và Linux.

Giao diện người dùng thân thiện: Visual Studio Code có một giao diện người dùng trực quan và dễ sử dụng, với các công cụ và tính năng được tổ chức một cách hợp lý.

Hỗ trợ nhiều ngôn ngữ lập trình: Visual Studio Code hỗ trợ nhiều ngôn ngữ lập trình khác nhau bao gồm HTML, CSS, JavaScript, TypeScript, Python, Java, PHP, và nhiều ngôn ngữ khác.

Mở rộng bằng Extensions: Bạn có thể mở rộng khả năng của Visual Studio Code thông qua các extension, cho phép bạn tùy chỉnh và thêm các tính năng mới như debug, linting, autocomplete, và nhiều chức năng khác.

IntelliSense: Visual Studio Code cung cấp IntelliSense, một tính năng tự động hoàn thành mã và gợi ý thông minh giúp tăng tốc quá trình viết mã.

Debugging và linting: Visual Studio Code cung cấp tính năng debugging tích hợp và hỗ trợ linting để giúp bạn xác định và sửa lỗi mã một cách dễ dàng.

Live Server: Extension Live Server cho phép bạn khởi chạy một máy chủ web cục bộ và tự động làm mới trang web của bạn khi bạn lưu các tệp mã nguồn.

Git integration: Visual Studio Code tích hợp tốt với Git, cho phép bạn quản lý phiên bản mã nguồn và thực hiện các thao tác Git như commit, push, pull trực tiếp từ giao diện người dùng.

Hỗ trợ nền tảng: Visual Studio Code hoạt động trên nhiều nền tảng như Windows, macOS và Linux, cho phép bạn làm việc trên bất kỳ máy tính nào mà không bị giới hạn.

Backend

❖ **Node.js**

Node.js là một môi trường chạy mã JavaScript ở phía máy chủ. Nó cho phép bạn xây dựng các ứng dụng web và máy chủ hiệu quả bằng cách sử dụng JavaScript, một ngôn ngữ chủ đạo của phía máy khách. [9]

Tính năng và ưu điểm: Bất đồng bộ, không chặn I/O, khả năng xử lý hàng nghìn kết nối đồng thời, thích hợp cho các ứng dụng thời gian thực.

❖ **Express**

Express.js là một thư viện (framework) ứng dụng web Node.js linh hoạt. Nó giúp đơn giản hóa việc xây dựng ứng dụng web và API Node.js. [10]

Tính năng và ưu điểm: Đơn giản, linh hoạt, middleware mạnh mẽ, tạo các định tuyến dễ dàng.

❖ **Body-Parser**

Body-Parser là một middleware cho Express.js giúp xử lý dữ liệu đến từ các yêu cầu HTTP (như POST và PUT requests).

Tính năng và ưu điểm: Giúp trích xuất dữ liệu từ phần chính của yêu cầu, xử lý dữ liệu định dạng JSON và dữ liệu trên form.

❖ **Dữ liệu**

OpenStreetMap (OSM): Là một dự án mã nguồn mở toàn cầu lớn nhất thế giới, được cộng đồng toàn cầu cập nhật và duy trì. Dữ liệu trong OSM bao gồm thông tin về đường phố, con đường, điểm quan trọng, dải phân cách, sông, hồ, rừng, công viên, và nhiều thông tin khác. Điều này làm cho OSM trở thành một nguồn tài nguyên quan trọng cho nhiều ứng dụng và dự án liên quan đến bản đồ và địa lý.

Overpass Turbo**:** Overpass Turbo là một công cụ trực tuyến mạnh mẽ được sử dụng để truy vấn và trích xuất dữ liệu từ OpenStreetMap (OSM). [11]

Truy vấn dữ liệu OSM: Overpass Turbo cho phép bạn tạo và thực thi các truy vấn OSM để tìm và trích xuất dữ liệu từ cơ sở dữ liệu OSM rất lớn. Tìm các đối tượng như điểm quan trọng, đường phố, công viên, nhà hàng, bệnh viện, trạm xăng ,... và nhiều loại đối tượng khác trên bản đồ.

Tính năng: Overpass Turbo cung cấp giao diện trực quan cho kết quả truy vấn, giúp bạn dễ dàng nhận biết và hiểu thông tin.

Lập trình: Bạn có thể sử dụng ngôn ngữ truy vấn Overpass QL (Overpass Query Language) để tạo các truy vấn phức tạp. Hỗ trợ lập trình JavaScript cho việc tùy chỉnh và xử lý kết quả truy vấn.

Xem trước kết quả: Overpass Turbo cho phép xem trước kết quả truy vấn trên bản đồ trực tuyến và trong bảng dữ liệu.

Xuất dữ liệu: Bạn có thể xuất kết quả truy vấn dưới dạng GeoJSON, JSON hoặc các định dạng khác như GPX, KML.

Sử dụng trực tiếp trên trình duyệt: Overpass Turbo hoạt động trực tiếp trên trình duyệt web mà không cần cài đặt bất kỳ phần mềm nào.

Địa chỉ website: https://overpass-turbo.eu

Dễ sử dụng: Giao diện đơn giản và dễ sử dụng, thích hợp cho cả người mới bắt đầu và người có kinh nghiệm.

❖ **File JSON (JavaScript Object Notation)**

JSON là một định dạng dữ liệu phổ biến được sử dụng để truyền tải và lưu trữ dữ liệu dưới dạng văn bản. JSON là một phần của JavaScript, nhưng nó cũng có thể được sử dụng bởi nhiều ngôn ngữ lập trình khác.

Cấu trúc cơ bản: JSON sử dụng cấu trúc đối tượng và mảng để tổ chức dữ liệu. [3]

Dễ đọc và hiểu: JSON sử dụng cú pháp đơn giản và gần gũi với con người, dễ đọc và hiểu.

Dễ dàng tích hợp với nhiều ngôn ngữ: JSON là định dạng dữ liệu độc lập với ngôn ngữ, nên có thể sử dụng với nhiều ngôn ngữ lập trình khác nhau.

Kích thước nhỏ gọn: JSON thường có kích thước nhỏ gọn so với các định dạng dữ liệu khác như XML, làm cho việc truyền tải dữ liệu trở nên hiệu quả hơn.

Lưu trữ dữ liệu: JSON thường được sử dụng để lưu trữ cấu trúc dữ liệu như cài đặt ứng dụng, cấu hình, hoặc dữ liệu từ máy chủ. [4]

Truyền tải dữ liệu: JSON được sử dụng làm định dạng dữ liệu phổ biến trong việc truyền tải dữ liệu giữa máy chủ và trình duyệt thông qua AJAX hoặc các API.

Làm việc với API: Hầu hết các API trả về dữ liệu dưới dạng JSON, cho phép các ứng dụng web hoặc di động tương tác với dữ liệu này một cách dễ dàng.

Frontend

❖ **HTML (HyperText Markup Language)**

HTML là ngôn ngữ đánh dấu sử dụng để tạo ra cấu trúc và nội dung của trang web.

Dễ học và sử dụng: HTML là một ngôn ngữ đơn giản và dễ học, rất thích hợp cho người mới bắt đầu.

Khả năng tương thích cao: Trang web viết bằng HTML có thể hiển thị trên nhiều trình duyệt và thiết bị khác nhau.

Thuận tiện cho tìm kiếm: HTML có thể tối ưu hóa dễ dàng cho các công cụ tìm kiếm, giúp trang web dễ dàng được tìm thấy và xếp hạng cao.

❖ **CSS (Cascading Style Sheets)**

CSS là ngôn ngữ được sử dụng để tạo kiểu cho các phần tử HTML, làm cho trang web trở nên đẹp mắt và dễ đọc.

Tách biệt cấu trúc và kiểu dáng: CSS giúp tách biệt cấu trúc và kiểu dáng, làm cho việc chỉnh sửa giao diện trở nên linh hoạt và dễ dàng.

Tính duyệt biến cao: CSS hỗ trợ nhiều tính năng duyệt biến như màu sắc, font chữ, margin, padding, giúp tạo ra giao diện đa dạng.

Tiết kiệm thời gian và hiệu quả: Bằng cách áp dụng CSS, bạn có thể thay đổi kiểu dáng trang web một cách nhanh chóng và dễ dàng.

❖ **JavaScript**

JavaScript là một ngôn ngữ lập trình mạnh mẽ được sử dụng chủ yếu để tạo ra các chức năng tương tác trên trang web.

Tính tương tác cao: JavaScript cho phép tạo ra các hiệu ứng tương tác như di chuyển, thêm/xóa phần tử HTML, kiểm tra dữ liệu nhập và nhiều hơn nữa.

Tính linh hoạt và mạnh mẽ: JavaScript có thể tích hợp với nhiều thư viện và framework khác nhau để phát triển ứng dụng web phức tạp.

Xử lý dữ liệu người dùng: JavaScript cho phép xử lý dữ liệu trên trình duyệt người dùng mà không cần tải lại trang.

❖ **Leaflet**

Leaflet là một thư viện JavaScript mã nguồn mở được sử dụng để tạo ra bản đồ tương tác trên các trang website. Leaflet là một trong những thư viện bản đồ phổ biến nhất và linh hoạt, giúp bạn tạo ra các ứng dụng bản đồ đa dạng với các tính năng tùy chỉnh. [12]

Nhẹ nhàng và linh hoạt: Leaflet là một thư viện nhẹ nhàng và dễ sử dụng, cho phép bạn tạo ra các bản đồ tương tác một cách nhanh chóng và hiệu quả.

Hỗ trợ đa nền tảng: Leaflet hoạt động tốt trên nhiều thiết bị và trình duyệt khác nhau, bao gồm cả điện thoại di động và máy tính bảng.

Dễ dàng tích hợp: Thư viện này cho phép tích hợp các dịch vụ bản đồ như OpenStreetMap, Mapbox, Bing, và các dịch vụ bản đồ khác một cách dễ dàng.

Tính tùy biến cao: Leaflet cung cấp nhiều tính năng tùy chỉnh, bao gồm điều chỉnh kiểu dáng của các lớp, thêm điểm đánh dấu, tạo hành trình, và thậm chí là tạo các lớp bản đồ từ dữ liệu JSON hoặc GeoJSON.

Hỗ trợ plugin phong phú: Cộng đồng Leaflet đã phát triển nhiều plugin mạnh mẽ cho các tính năng như hiển thị dữ liệu đa phương tiện, điều hướng, đo lường.

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2.1. Phân tích bài toán

Từ nội dung bài toán đã được phát biểu và các yêu cầu chức năng đã đề cập ở chương 1, hệ thống website tìm đường đi ngắn nhất được phân tích chi tiết như sau:

Xác định yêu cầu

- ☞ Tìm đường đi: Website cần cho phép người dùng chọn phương tiện di chuyển, chọn hai điểm trên bản đồ và tìm ra đường đi ngắn nhất giữa chúng.
- ☞ Hiển thị bản đồ: Website cần hiển thị một bản đồ để người dùng có thể chọn phương tiện di chuyển, điểm xuất phát và điểm đích.

Thu thập dữ liệu

- ☞ Dữ liệu bản đồ: Cần sử dụng dữ liệu bản đồ từ một nguồn bản đồ như OpenStreetMap.

Xử lý dữ liệu

- ☞ Tính toán đường đi: Cần sử dụng thuật toán đường đi ngắn nhất để tính toán đường đi từ điểm xuất phát đến điểm đích trên bản đồ.

Thiết kế giao diện

- ☞ Trang chủ: Hiển thị bản đồ và giao diện để nhập phương tiện di chuyển, điểm xuất phát và điểm đích.
- ☞ Kết quả: Hiển thị đường đi ngắn nhất từ điểm xuất phát đến điểm đích.

Phát triển website

- ☞ Frontend: Sử dụng HTML, CSS và JavaScript để xây dựng giao diện trang web và tương tác với người dùng. [1][2]
- ☞ Backend: Sử dụng ngôn ngữ lập trình như JavaScript (Node.js) để xử lý các yêu cầu từ frontend, tính toán đường đi và trả về kết quả cho người dùng.

Kiểm thử và sửa lỗi

- ☞ Kiểm thử tính năng: Kiểm tra tính đúng đắn và hiệu suất của tính năng tìm đường đi ngắn nhất.

☞ Kiểm thử giao diện: Đảm bảo giao diện hoạt động một cách mượt mà và thân thiện với người dùng.

Triển khai

☞ Triển khai website: Đưa website vào hoạt động trên một máy chủ và công bố cho người dùng sử dụng.

Các yếu tố quan trọng

☞ Tính tích hợp: Website cần tích hợp thuật toán đường đi ngắn nhất và hiển thị bản đồ một cách mạch lạc và hiệu quả.

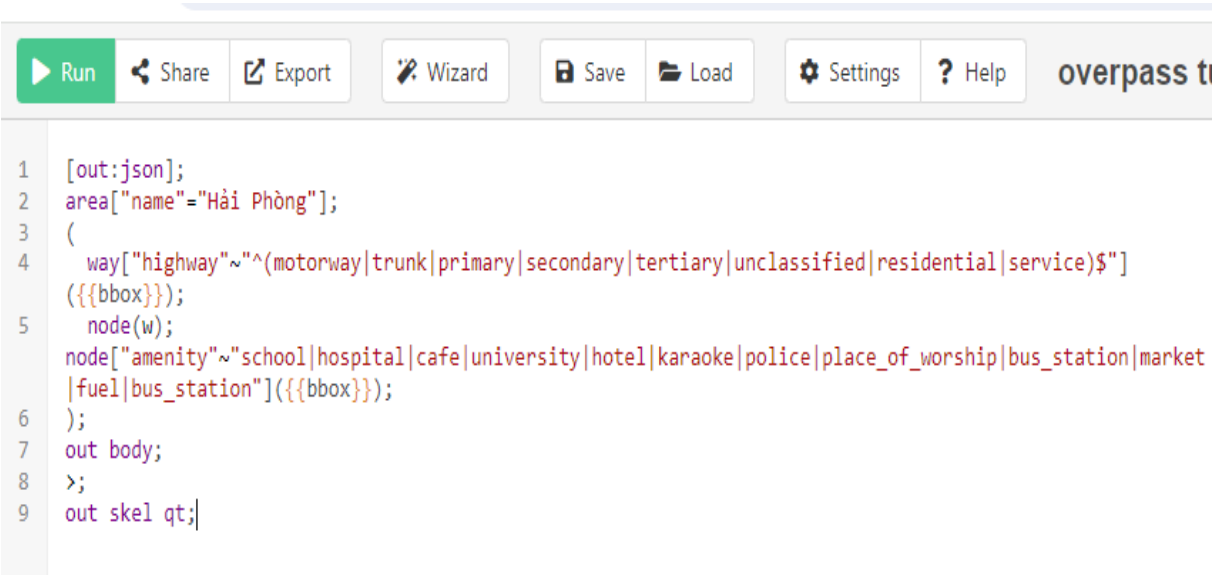
☞ Tính tương tác: Giao diện cần tương tác tốt với người dùng, cho phép họ dễ dàng nhập điểm xuất phát và điểm đích và nhận kết quả một cách nhanh chóng.

2.2. Phương pháp lưu trữ dữ liệu của bản đồ

Tải dữ liệu bản đồ

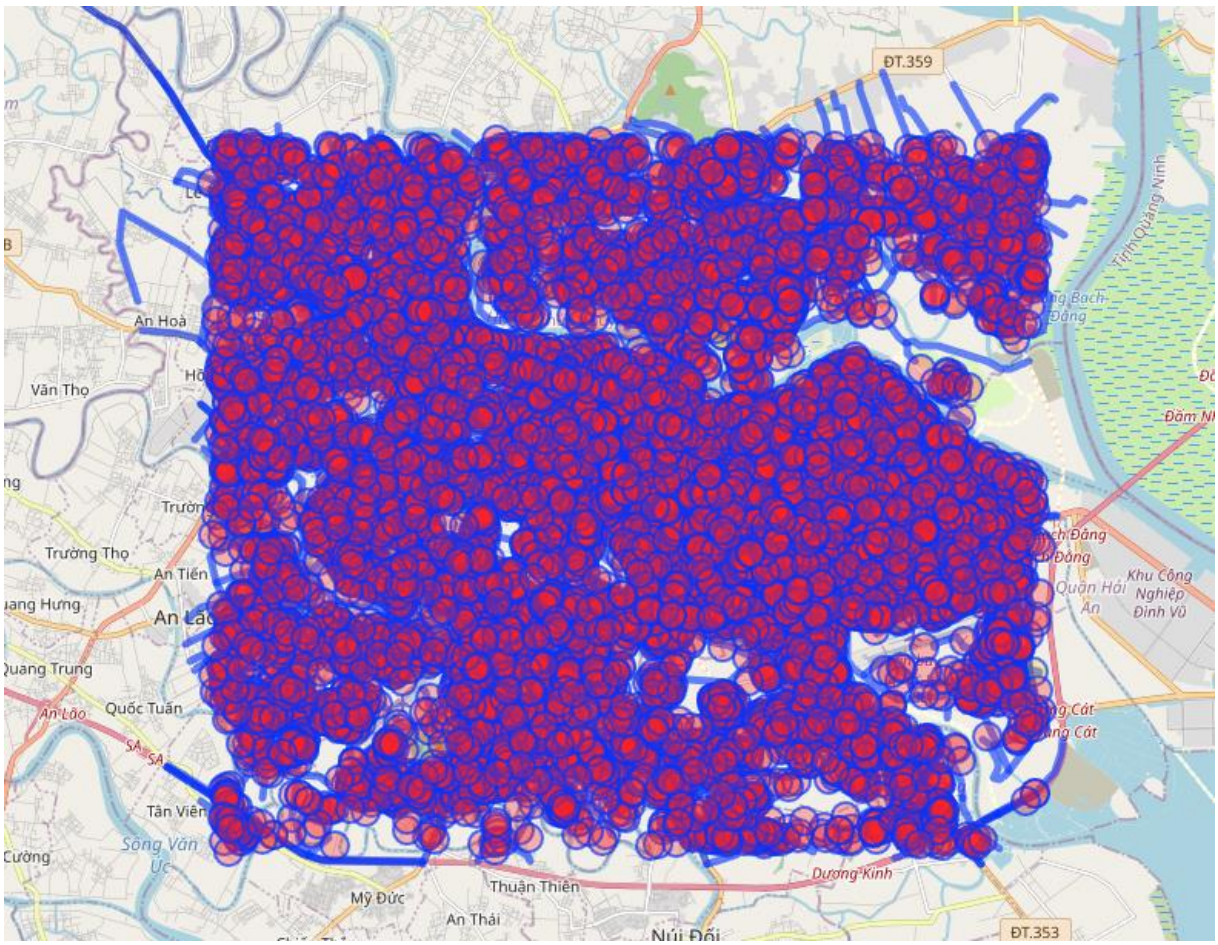
+B1: Truy cập trang web https://overpass-turbo.eu.

+B2: Sử dụng ô văn bản để tạo truy vấn OSM bằng ngôn ngữ truy vấn Overpass QL.



Hình 2.1: Các câu truy vấn lấy dự liệu từ website http://overpass-turbo.edu

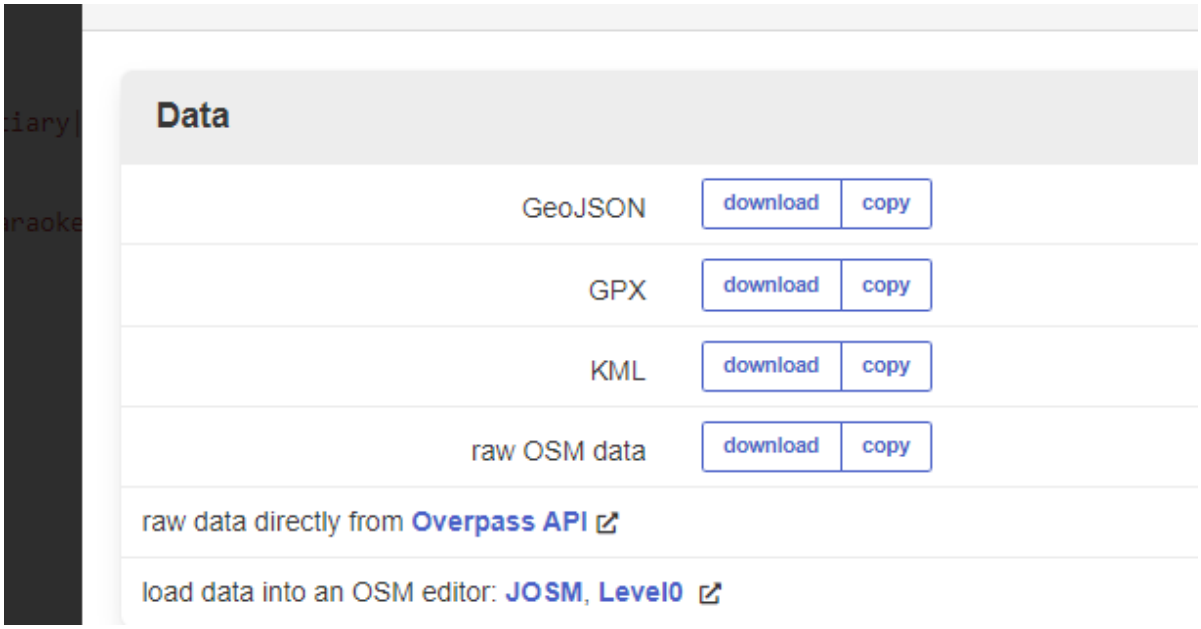
+B3: Nhấn nút "Run" để thực thi truy vấn và xem kết quả trên bản đồ.



Hình 2.2: Phạm vi dữ liệu được tải về

Có thể điều chỉnh truy vấn và thực thi lại để tinh chỉnh kết quả.

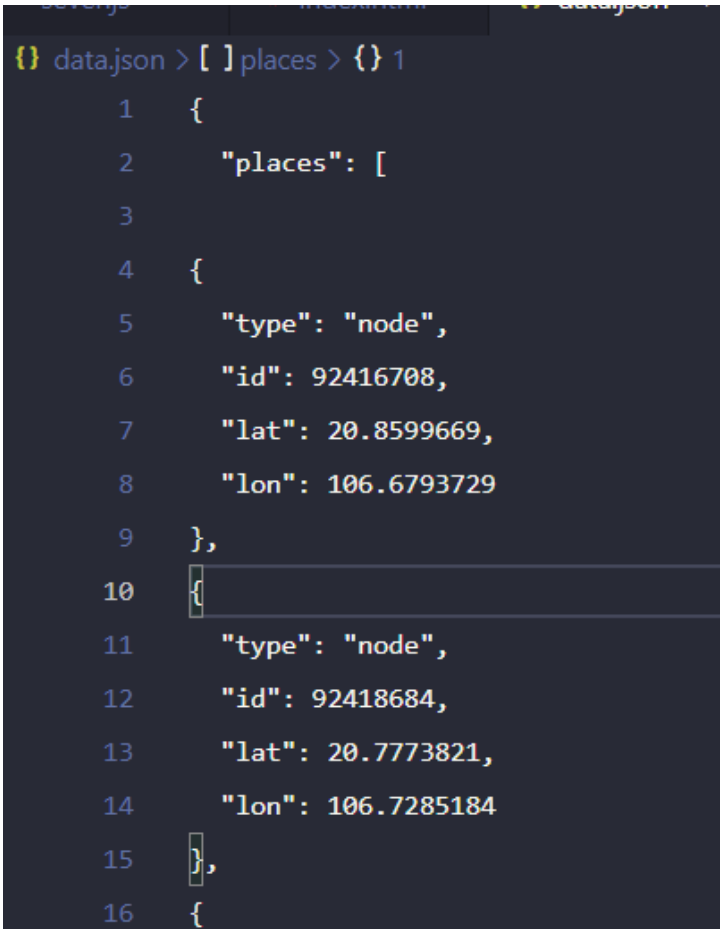
+B4: Khi kết quả phù hợp, bạn có thể xuất dữ liệu sang các định dạng khác nhau.



Hình 2.3: Các tùy chọn định dạng dữ liệu tải về

- Lưu trữ dữ liệu

☞ Kết quả dữ liệu bản đồ tải về trong file data.json



Hình 2.4: Dữ liệu tải về được lưu dưới dạng file json

- Mô tả dữ liệu bản đồ

☞ Thông tin dữ liệu bản đồ data: Dữ liệu bản đồ bao gồm 20,104 điểm (node) và 1,883 đường (ways). Dữ liệu lưu trữ dưới dạng JSON với dung lượng là 2.25MB.

☞ Phạm vi dữ liệu bản đồ: Dữ liệu bản đồ bao gồm các nodes và ways. Các node đại diện cho các điểm trên bản đồ với thông tin về vĩ độ (latitude) và kinh độ (longitude). Các ways là tập hợp các node để tạo thành các đoạn đường hoặc các phần tử không gian khác trên bản đồ.

☞ Cấu trúc dữ liệu: Dữ liệu được lưu cấu trúc đối tượng gồm key và value.

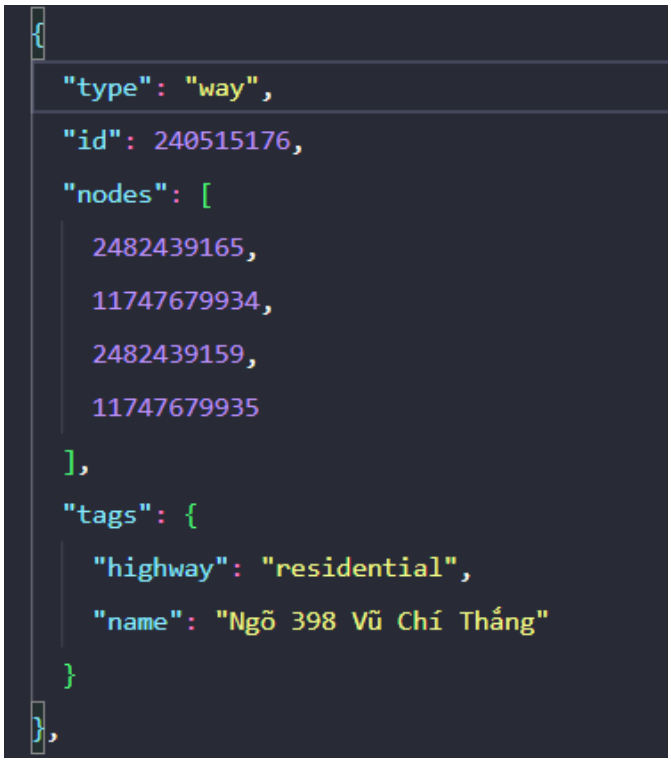
- Node (điểm)

- type: Loại của đối tượng, trong trường hợp này là "node". Đây là một chuỗi ký tự chỉ ra loại của đối tượng (node, way).
- id: Một số duy nhất đại diện cho node trong hệ thống OSM.
- lat: Tọa độ vĩ độ của node, được biểu diễn dưới dạng số thập phân.

- lon: Tọa độ kinh độ của node, cũng được biểu diễn dưới dạng số thập phân.

Hình 2.5: Dữ liệu một điểm trong file data

- Way (đường)
  - type: Loại của đối tượng, trong trường hợp này là "way".
  - id: Một số duy nhất đại diện cho way trong hệ thống OSM.
  - nodes: Một mảng các số duy nhất đại diện cho các node mà way này đi qua.
  - tags: Một tập hợp các cặp key-value mô tả các thuộc tính của way, chẳng hạn như thông tin về loại đường, tốc độ tối đa, tên đường, v.v.

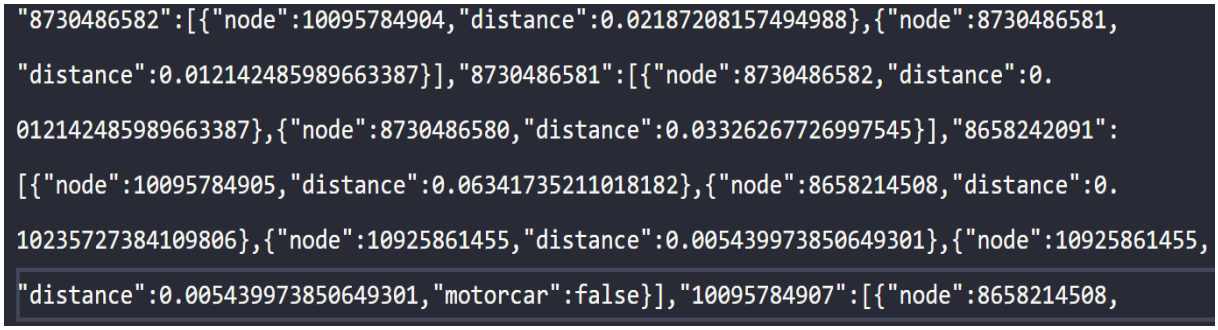


Hình 2.6: Dữ liệu một đoạn đường trong file data

Trong hình trên: Tuyến đường có id là 240515176 đi qua các node có id lần lượt là 2482439165, 11747679934, 2482439159 và 11747679935.Các tags

mô tả rằng đây là một đường dân cư ("highway": "residential") tên đường là " Ngõ 398 Vũ Chí Thắng".

Mô tả dữ liệu đồ thị graph



"8730486582":[{"node":10095784904,"distance":0.02187208157494988},{"node":8730486581,
"distance":0.012142485989663387}],"8730486581":[{"node":8730486582,"distance":0.
012142485989663387},{"node":8730486580,"distance":0.03326267726997545}],"8658242091":
[{"node":10095784905,"distance":0.06341735211018182},{"node":8658214508,"distance":0.
10235727384109806},{"node":10925861455,"distance":0.005439973850649301},{"node":10925861455,
"distance":0.005439973850649301,"motorcar":false}],"10095784907":[{"node":8658214508,

Hình 2.7: Dữ liệu trong đồ thị graph

- Dữ liệu của đồ thị graph được tạo bởi các thông tin từ file data.json sau khi tính toán về khoảng cách giữa các điểm (node) trên đoạn đường, các đỉnh kề các thuộc tính như oneway, motorcar, motorcycle, foot có kiểu dự liệu Boolean có giá trị là true hoặc false.
- Cách tính khoảng cách sử dụng tọa độ địa lý: Khoảng cách giữa hai địa điểm có thể được tính bằng cách sử dụng tọa độ địa lý của chúng. Để tính khoảng cách này, ta cần biết địa điểm cụ thể của hai điểm trong hệ tọa độ địa lý. Hệ tọa độ địa lý thường là hệ tọa độ dựa trên Kinh độ và Vĩ độ, trong đó Kinh độ là độ dài theo chiều đông-tây và Vĩ độ là độ cao theo chiều bắc-nam. Công thức Haversine là một công thức phổ biến được sử dụng để ước tính khoảng cách giữa hai điểm trên mặt cầu. Công thức này dựa trên địa trị hình học và sử dụng bán kính của Trái Đất để tính toán khoảng cách. [13]
- Công thức Haversine được biểu diễn như sau:

Distance =

$2*r*\arcsin(\sqrt{\text{haversin}(\Delta lat) + \cos(lat1) * \cos(\text{lat2}) * \text{haversin}(\Delta\text{lon})})$

Trong đó: Δlat = lat2 – lat1, Δlon = lon2 – lon1.

$\text{haversin}(\Delta lat) = \sin^2(\frac{\Delta lat}{2})$, $\text{haversin}(\Delta\text{lon}) = \sin^2(\frac{\Delta lon}{2})$.

- r: Là bán kính trái đất.
- lat1, lon1 : Là vĩ độ và kinh độ điểm thứ nhất.
- lat2, lon2 : Là vĩ độ và kinh độ điểm thứ hai.
- Distance: Khoảng cách giữa hai điểm.

- Thông tin dữ liệu đồ thị graph: Dữ liệu bản đồ bao gồm 9944 đỉnh (node) tệp dữ liệu lưu trữ dưới dạng JSON với dung lượng là 2.25 MB.

- Cấu trúc dữ liệu: Dữ liệu đồ thị được biểu diễn dưới dạng một mảng, trong đó mỗi phần tử của mảng đại diện cho một điểm (node) trên đồ thị. Mỗi phần tử bao gồm một id của điểm và danh sách các đỉnh kề của điểm đó.

- Ví dụ hình 2.7: điểm có id là 8730486582 có các đỉnh kề là 10095784904 với khoảng cách tới điểm đó là 0.02187208157494988 kilomet, và đỉnh kề 8730486581 với khoảng cách tới điểm đó là 0.12142485989663387 kilomet.

2.3. Thuật toán tìm đường đi ngắn nhất

Hiện nay, việc tìm đường đi ngắn nhất vẫn là một lĩnh vực nghiên được nhiều sự quan tâm, với nhiều phát triển mới và sự tiến bộ trong các thuật toán và phương pháp. Bảng dưới đây tóm tắt về đặc điểm của một số thuật toán tìm đường đi ngắn nhất.

Bảng 2.1: Ưu nhược điểm của các thuật toán tìm đường đi ngắn nhất

Thuật toán	**Ưu điểm**	**Nhược điểm**
Thuật Toán Dijkstra	Dễ hiểu và triển khai. Hoạt động tốt trên đồ thị vô hướng và có trọng số dương, cho kết quả chính xác.	Không xử lý được trọng số âm. Độ phức tạp thời gian lớn (O((V+E)logV)) với V là số đỉnh và E là số cạnh.
Thuật Toán Bellman-Ford	Xử lý được trọng số âm và phát hiện chu trình âm. Hoạt động tốt trên đồ thị có trọng số âm.	Độ phức tạp thời gian lớn (O(VE)) với V là số đỉnh và E là số cạnh. Nếu có nhiều cạnh thuật toán trở lên chậm.
Thuật Toán A* (A Star):	Hiệu quả trên các đồ thị lớn. Sử dụng heuristics để tối ưu hóa tìm kiếm. Tính chính xác cao khi được áp dụng đúng cách.	Độ phức tạp thời gian phụ thuộc nhiều vào chất lượng của hàm heuristics. Cần phải chọn heuristics phù hợp.
Thuật Toán	Xử lý được cả đồ thị có trọng	Độ phức tạp thời gian lớn

Floyd-Warshall	số âm. Tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh.	($O(v^3)$) với V là số đỉnh. Yêu cầu lưu trữ ma trận trọng số, tốn bộ nhớ.
Thuật Toán Contraction Hierarchies	Hiệu quả trên các đồ thị lớn. Tối ưu hóa thời gian tìm kiếm bằng cách sử dụng các đỉnh "chia" và "trộn".	Yêu cầu tiền xử lý đồ thị phức tạp. Khó triển khai.
Thuật Toán Johnson	Xử lý được cả trọng số âm. Tìm kiếm đường đi ngắn nhất giữa tất cả các cặp đỉnh.	Độ phức tạp thời gian cao ($O(VE + v^2 \log V)$).
Thuật Toán Yen's K Shortest Paths	Tìm ra nhiều lựa chọn đường đi ngắn nhất giữa hai đỉnh. Phù hợp cho việc tìm kiếm đa mục tiêu.	Độ phức tạp thời gian cao khi cần tìm nhiều lựa chọn ($O(K(E + v^2 \log V))$ với K là số lựa chọn.
Thuật Toán SPFA (Shortest Path Faster Algorithm)	Xử lý được trọng số âm. Độ phức tạp thời gian thấp trong một số trường hợp.	Dễ bị lặp vô hạn trên các đồ thị có chu trình âm. Hiệu suất không ổn định trên các trường hợp tồi nhất.

Kết quả so sánh hiệu suất một số các thuật toán tìm đường ngắn nhất

Với phạm vi dữ liệu nhỏ

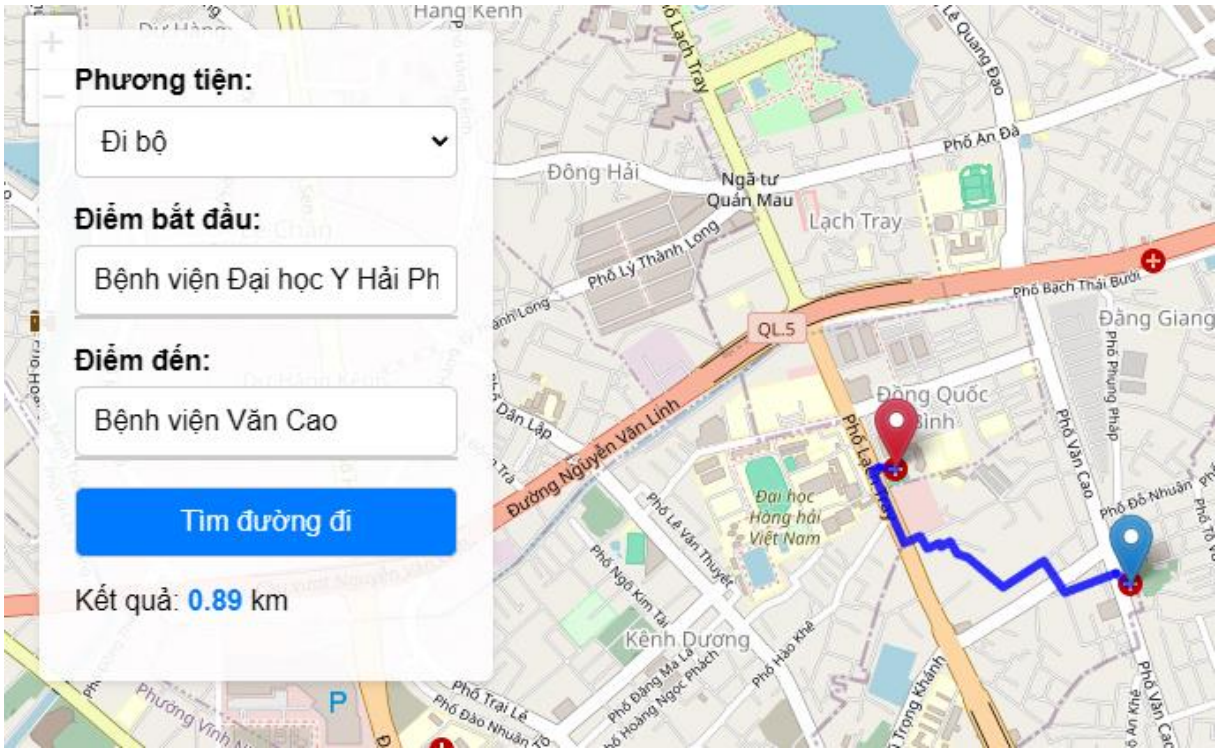
```
const graphtest = {
    "0":[{node: 1, distance: 2.5},{node: 2, distance: 2.0,"motorcar": false},{node: 3, distance: 2.1}],
    "1":[{node: 0, distance: 2.5},{node: 4, distance: 1.0,"motorcycle": false}],
    "2":[{node: 0, distance: 2.0,"motorcar": false},{node: 4, distance:0.6},{node: 5, distance: 1.5}],
    "3":[{node: 0, distance: 2.1},{node: 5, distance:2.5}],
    "4":[{node: 1, distance: 1.0, "motorcycle": false},{node: 2, distance:0.6},{node: 6, distance:2.3}],
    "5":[{node: 2, distance: 1.5},{node: 3, distance:2.5},{node: 6, distance:1.9},{node: 7, distance:2.0}],
    "6":[{node: 4, distance: 2.3},{node: 5, distance:1.9},{node: 7, distance:1.8},{node: 8, distance:1.7}],
    "7":[{node: 5, distance: 2.0},{node: 6, distance:1.8},{node: 8, distance:2.0}],
    "8":[{node: 6, distance: 1.7},{node: 7, distance:2.0}]
}
```

Hình 2.8: Dữ liệu trong đồ thị kiểm tra graphtest

```
Graph file exists and loaded successfully.
Dijkstra:
Total distance: 5.5
Shortest path found: [ 0, 2, 5, 7 ]
Dijkstra Execution Time: 2.215ms
Bellman-Ford:
Total distance: 5.5
Shortest path found: [ 0, 2, 5, 7 ]
Bellman-Ford Execution Time: 0.446ms
Floyd-Warshall:
shortestPath [ 0, 2, 5, 7 ]
totalPathDistance 5.5
Floyd-Warshall Execution Time: 0.615ms
```

Hình 2.9: Kết quả kiểm tra thuật toán Dijkstra, Bellman-Ford và Floyd-Warshall với dữ liệu graphtest

Với phạm vi dữ liệu của dự án



Hình 2.10: Chạy thử chương trình với dữ liệu dự án

```
Dijkstra Execution Time: 177.895ms
Toltal path distance:  0.8928026858369155

 Floyd-Warshall Execution Time: 1:46:26.186 (h:mm:ss.mmm)

Bellman-Ford Execution Time: 1:56.841 (m:ss.mmm)
Toltal path distance:  0.8928026858369155
```

Hình 2.11: Kết quả kiểm tra thuật toán Dijkstra, Bellman-Ford và Floyd-Warshall với dữ liệu dự án

So sánh: Kết quả thử nghiệm trên các bộ dữ liệu nhỏ và lớn cho thấy sự khác biệt rõ rệt về thời gian thực thi giữa các thuật toán:

Dijkstra: 2.215 ms (dữ liệu nhỏ), 177.895 ms (dữ liệu lớn).

Bellman-Ford: 0.446 ms (dữ liệu nhỏ), 1:56.841 (m:ss.mmm) (dữ liệu lớn).

Floyd-Warshall: 0.615 ms (dữ liệu nhỏ), 1:46:26.186 (h:m:ss.mmm) (dữ liệu lớn).

Với dữ liệu lớn, thời gian chạy của Dijkstra (xấp xỉ 177 mili giây) là ngắn hơn rất nhiều so với Bellman-Ford (xấp xỉ 1 phút 56 giây) và đặc biệt là Floyd-Warshall (xấp xỉ 1 giờ 46 phút 26 giây).

Tổng quãng đường tìm được và đường đi hoàn toàn khớp với kết quả của Bellman-Ford và Floyd-Warshall, chứng minh tính chính xác của Dijkstra.

Vì vậy: Xây dựng một ứng dụng web tìm đường đi ngắn nhất trên bản đồ với dữ liệu từ OSM trong khu vực nhỏ của một thành phố việc sử dụng Dijkstra vẫn là một lựa chọn tốt với các lý do sau:

- Đồ thị không có trọng số âm: Hoạt động hiệu quả trên những đồ thị không có trọng số âm.
- Hiệu quả trên đồ thị nhỏ và mật độ thấp: Hoạt động tốt trên các đồ thị có số lượng đỉnh nhỏ và mật độ cạnh thấp, điều này phù hợp với việc xây dựng một ứng dụng tìm đường trên một khu vực nhỏ của thành phố.
- Dễ triển khai: Dễ triển khai và hiểu, đặc biệt là khi bạn chỉ cần tính toán trên một phạm vi nhỏ và không cần xử lý trọng số âm.
- Tính chính xác: Đảm bảo tìm ra đường đi ngắn nhất giữa hai điểm trên đồ thị không có trọng số âm.
- Tính linh hoạt: Có thể được điều chỉnh để xử lý các biến thể của vấn đề, như tìm đường đi cho một loại phương tiện cụ thể (ví dụ: đường chỉ dành cho người đi bộ, ô tô, xe máy …).

Giới thiệu thuật toán Dijkstra

- Thuật toán Dijkstra được phát triển bởi nhà toán học và nhà máy tính Hà Lan Edsger W. Dijkstra vào năm 1956 và được công bố lần đầu tiên trong bài báo khoa học "A Note on Two Problems in Connexion with Graphs" vào năm 1959. [5]

- Mục đích: Dijkstra phát triển thuật toán để tìm đường đi ngắn nhất giữa hai đỉnh trên một đồ thị có trọng số không âm.

- Phát triển và ứng dụng: Thuật toán Dijkstra nhanh chóng trở thành một trong những công cụ cơ bản và quan trọng trong lĩnh vực khoa học máy tính và các ứng dụng thực tiễn. Nó đã được sử dụng rộng rãi trong nhiều lĩnh vực như định tuyến mạng lưới, hệ thống điều hành tàu, máy bay, giao thông, logistics, truyền thông, và nhiều lĩnh vực khác.

Phương pháp

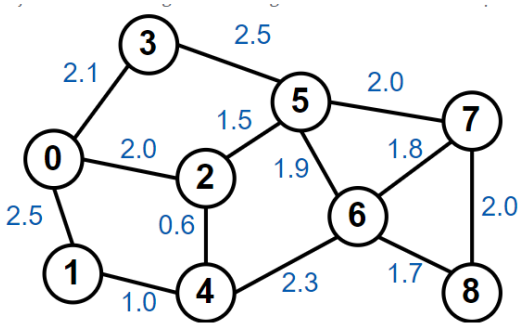
- Bước 1: Từ đỉnh gốc, khởi tạo khoảng cách tới chính nó là 0, khởi tạo khoảng cách nhỏ nhất ban đầu tới các đỉnh khác là +∞. Ta được danh sách các khoảng cách tới các đỉnh.

- Bước 2: Chọn đỉnh a có khoảng cách nhỏ nhất trong danh sách này và ghi nhận. Các lần sau sẽ không xét tới đỉnh này nữa.

- Bước 3: Lần lượt xét các đỉnh kề b của đỉnh a. Nếu khoảng cách từ đỉnh gốc tới đỉnh b nhỏ hơn khoảng cách hiện tại đang được ghi nhận thì cập nhật giá trị và đỉnh kề a vào khoảng cách hiện tại của b.

- Bước 4: Sau khi xét tất cả đỉnh kề b của đỉnh a. Lúc này ta được danh sách khoảng cách tới các điểm đã được cập nhật. Quay lại Bước 2 với danh sách này. Thuật toán kết thúc khi chọn được khoảng cách nhỏ nhất từ tất cả các điểm.

❖ **Ví dụ**



Hình 2.12: Đồ thị G

Thuật toán Dijkstra sẽ tìm khoảng cách từ đỉnh gốc 0 tới tất cả các đỉnh còn lại trong đồ thị G. Đầu tiên, khởi tạo khoảng cách nhỏ nhất ban đầu tới các đỉnh khác là +∞ và khoảng cách tới đỉnh gốc là 0. Ta được danh sách các khoảng cách tới các đỉnh.

Bảng 2.2: Khởi tạo khoảng cách ban đầu cho các đỉnh

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)

Chọn đỉnh 0 có giá trị nhỏ nhất, xét các đỉnh kề của đỉnh 0: Xét đỉnh 1, khoảng cách từ gốc đến đỉnh 1 là 2.5 < +∞ nên ghi nhận giá trị mới là (2.5,0) (nghĩa là khoảng cách đến đỉnh gốc hiện tại ghi nhận là 2.5, đỉnh kề liền trước là đỉnh 0). Xét tương tự cho đỉnh 2 và 3, ta được dòng thứ 2 trong bảng.

Bảng 2.3: Xét từ đỉnh gốc 0 đến đỉnh 3

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)

Sau khi xét tất cả các đỉnh ta chọn đỉnh 2 có khoảng cách nhỏ nhất và ghi nhận để xét tiếp. Tiếp tục xét đỉnh kề của 2 là đỉnh 4 và 5 với nguyên tắc nêu ở trên. Xét đỉnh 4, khoảng cách từ đỉnh gốc đến đỉnh 4 sẽ bằng khoảng cách từ đỉnh gốc tới đỉnh 2 cộng khoảng cách từ 2 đến 4. Nghĩa là 2.0+0.6=2.6 nên ta ghi nhận khoảng cách tại đỉnh 4 là (2.6,2). Xét tương tự cho đỉnh 5.

Bảng 2.4: Xét từ đỉnh gốc 0 đến đỉnh 5

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	(2.1,0)	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)

Lúc này ta chọn được đỉnh 3 có khoảng cách nhỏ nhất, xét đỉnh kề của đỉnh 3 là đỉnh 5. Khoảng cách từ gốc tới đỉnh 5 =2.1+2.5=4.6. Lớn hơn khoảng cách hiện tại được ghi nhận, vì vậy giá trị tại đỉnh 5 không đổi.

Bảng 2.5: Xét từ đỉnh gốc 0 đến đỉnh 5

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	(2.1,0)	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)

Đỉnh 1 là đỉnh được chọn tiếp theo, xét đỉnh kề của 1 là đỉnh 4. Khoảng cách từ đỉnh gốc không nhỏ hơn khoảng cách hiện tại nên ta không cập nhật gì ở đỉnh này.

Bảng 2.6: Xét từ đỉnh gốc 0 đến đỉnh 5

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	(2.1,0)	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)

Sau khi xét xong ta chọn được đỉnh 4 là đỉnh tiếp theo. Ta cập nhật giá trị mới cho đỉnh 6.

Bảng 2.7: Xét từ đỉnh gốc 0 đến đỉnh 6

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	(2.1,0)	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	-	(3.5,2)	(4.9,4)	(∞,-)	(∞,-)

Chọn được đỉnh 5 là đỉnh nhỏ nhất, tiếp tục xét các đỉnh kề.

Bảng 2.8: Xét từ đỉnh gốc 0 đến đỉnh 7

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	(2.1,0)	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	-	(3.5,2)	(4.9,4)	(∞,-)	(∞,-)
-	-	-	-	-	-	(4.9,4)	(5.5,5)	(∞,-)

Đỉnh 6 là đỉnh tiếp theo được chọn.

Bảng 2.9: Xét từ đỉnh gốc 0 đến đỉnh 8

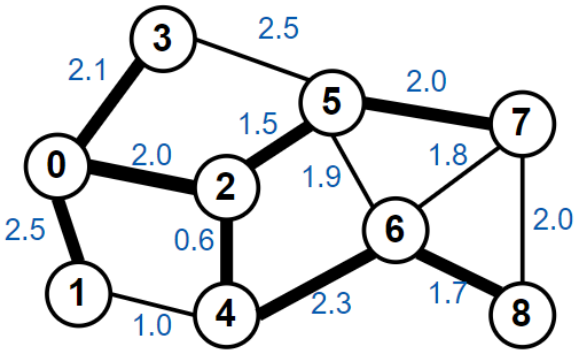
0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	(2.1,0)	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	-	(3.5,2)	(4.9,4)	(∞,-)	(∞,-)
-	-	-	-	-	-	(4.9,4)	(5.5,5)	(∞,-)
-	-	-	-	-	-	-	(5.5,5)	(6.6,6)

Chọn đỉnh có khoảng cách nhỏ nhất là đỉnh 7.

Bảng 2.10: Đỉnh nhỏ nhất từ đỉnh gốc 0 tới đỉnh 8

0	1	2	3	4	5	6	7	8
0	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	(2.0,0)	(2.1,0)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	(2.1,0)	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	(2.5,0)	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	(2.6,2)	(3.5,2)	(∞,-)	(∞,-)	(∞,-)
-	-	-	-	-	(3.5,2)	(4.9,4)	(∞,-)	(∞,-)
-	-	-	-	-	-	(4.9,4)	(5.5,5)	(∞,-)
-	-	-	-	-	-	-	(5.5,5)	(6.6,6)
-	-	-	-	-	-	-	-	(6.6,6)

Thuật toán kết thúc khi chọn được khoảng cách nhỏ nhất cho tất cả các đỉnh.



Hình 2.13: *Đồ thị G*

Hiệu năng:

Trên đồ thị với V đỉnh và E cạnh, thời gian thực thi của thuật toán Dijkstra là: O ((V+E)logV).

Ứng dụng thực tế của thuật toán Dijkstra

- Ứng dụng bản đồ trực tuyến: Trong các ứng dụng bản đồ trực tuyến như Google Maps, Apple Maps, hay Waze, Dijkstra được sử dụng để tìm đường đi ngắn nhất giữa hai địa điểm. Khi bạn nhập vào điểm xuất phát và điểm đích, hệ thống sẽ sử dụng Dijkstra để tính toán và hiển thị đường đi ngắn nhất.

- Hệ thống giao hàng và logistics: Trong hệ thống giao hàng và logistics, Dijkstra được sử dụng để tìm đường đi ngắn nhất giữa các điểm giao hàng và các điểm đích. Điều này giúp tối ưu hóa tuyến đường và giảm thời gian giao hàng cũng như chi phí vận chuyển.

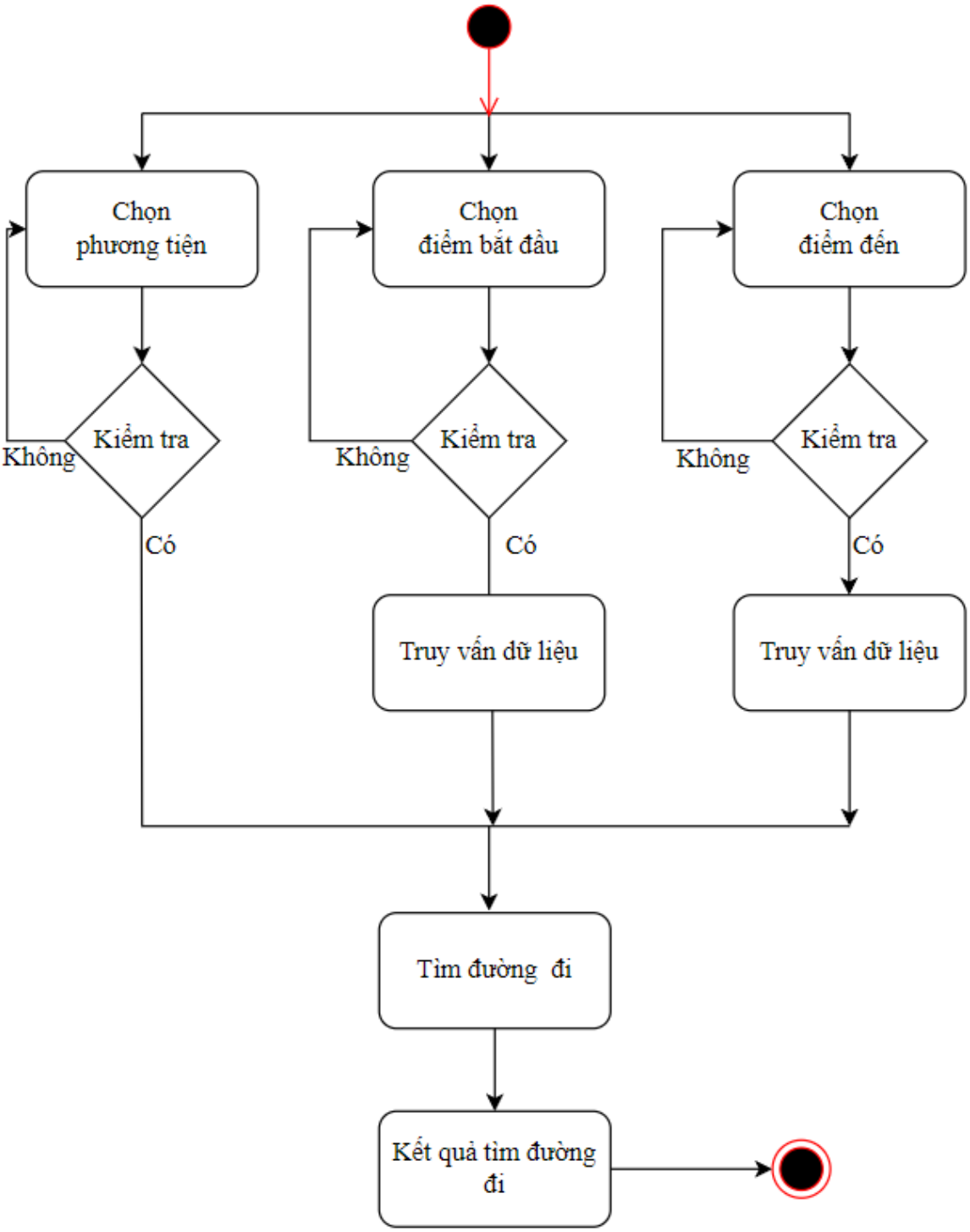
- Hệ thống điều hành tàu, máy bay: Trong hệ thống điều hành tàu, máy bay, Dijkstra được sử dụng để tính toán các tuyến đường an toàn và hiệu quả. Nó giúp xác định đường đi ngắn nhất và tối ưu nhất từ điểm khởi hành đến điểm đích, đồng thời đảm bảo an toàn và tiết kiệm nhiên liệu.

- Mạng lưới truyền thông: Trong mạng lưới truyền thông, Dijkstra được sử dụng để định tuyến dữ liệu từ một nguồn tới một đích thông qua đường đi ngắn nhất. Điều này giúp tối ưu hóa đường truyền dữ liệu và giảm độ trễ trong truyền thông.

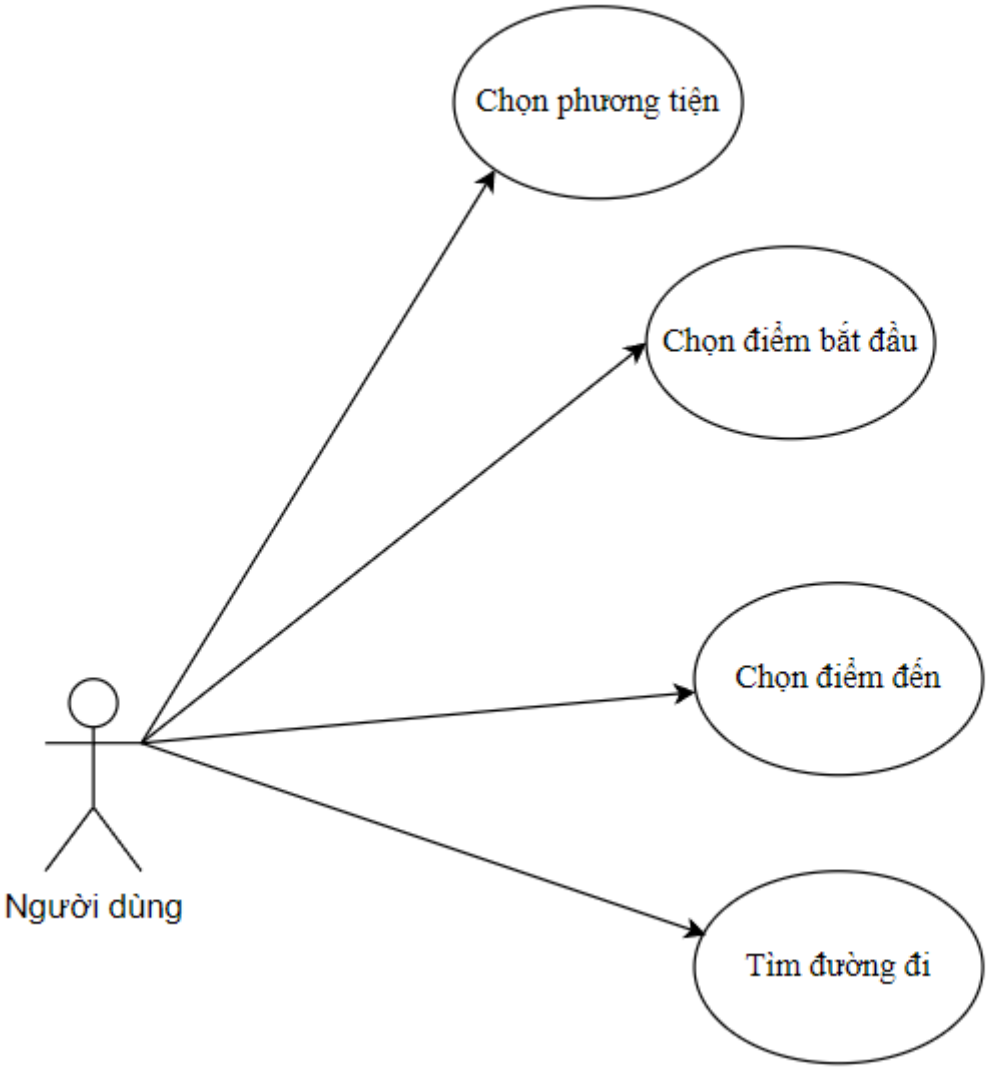
2.4. Biểu đồ Use Case

Bảng 2.11: Bảng đặc tả Use Case "Tìm đường đi ngắn nhất"

Tên Use Case	Tìm đường đi ngắn nhất
Chức năng	Tìm đường đi ngắn nhất giữa hai điểm và phương tiện di chuyển đã được chọn từ người dùng.
Tác nhân chính	Người dùng.
Điều kiện trước	Không có.
Điều kiện sau	Hiển thị thành công kết quả tìm đường đi.
Mô tả	Người dùng cung cấp thông tin như phương tiện, điểm bắt đầu và điểm đến để tìm đường đi ngắn nhất.
Kích hoạt	Khi người dùng nhấn vào nút "Tìm đường đi"
Chuỗi sự kiện chính: 1. Người dùng nhập hoặc chọn các thông tin phương tiện, điểm bắt đầu và điểm đến. 2. Hệ thống kiểm tra và tính toán đường đi ngắn nhất. 3. Hệ thống hiển thị kết quả tìm đường đi cho người dùng.	
Ngoại lệ: 1. Nếu người dùng không chọn phương tiện, điểm bắt đầu hoặc điểm đến, hệ thống sẽ thông báo và yêu cầu người dùng chọn lại thông tin. 2. Người dùng có thể nhập hoặc chọn lại thông tin nếu cần thiết.	

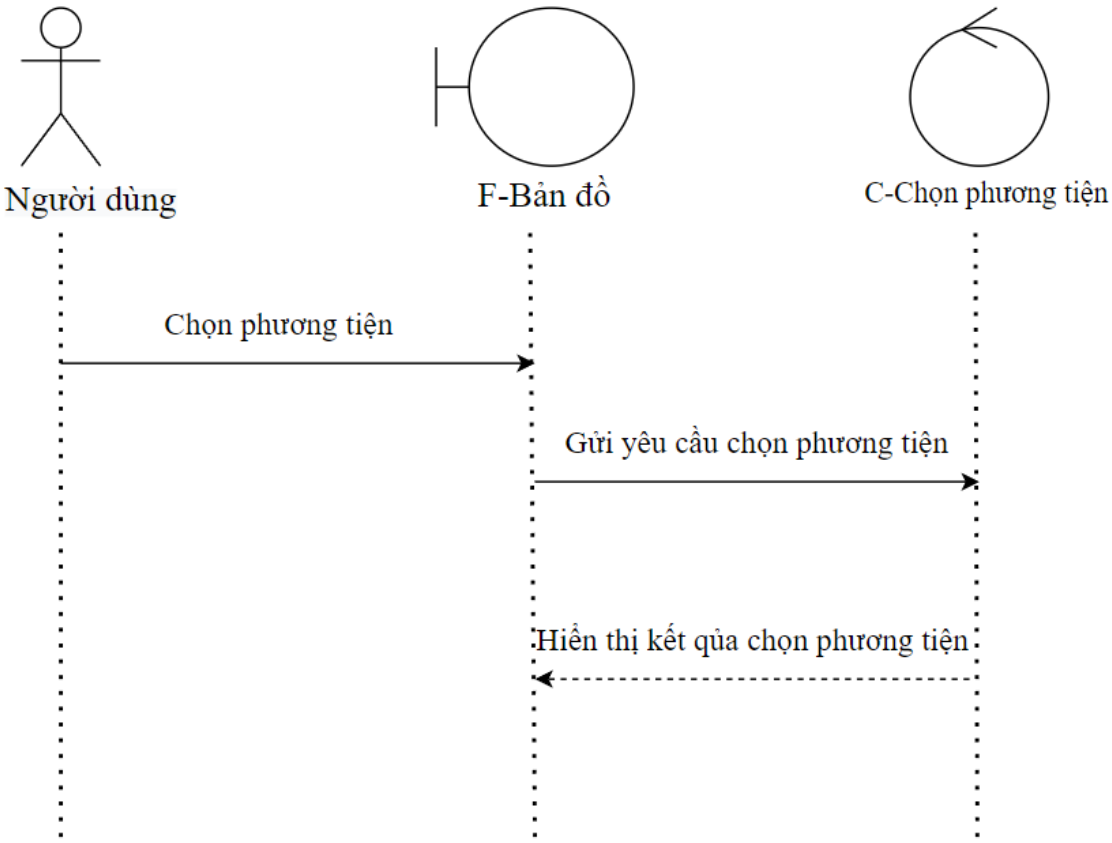


Hình 2.14: Sơ đồ nghiệp vụ “Tìm đường đi ngắn nhất”

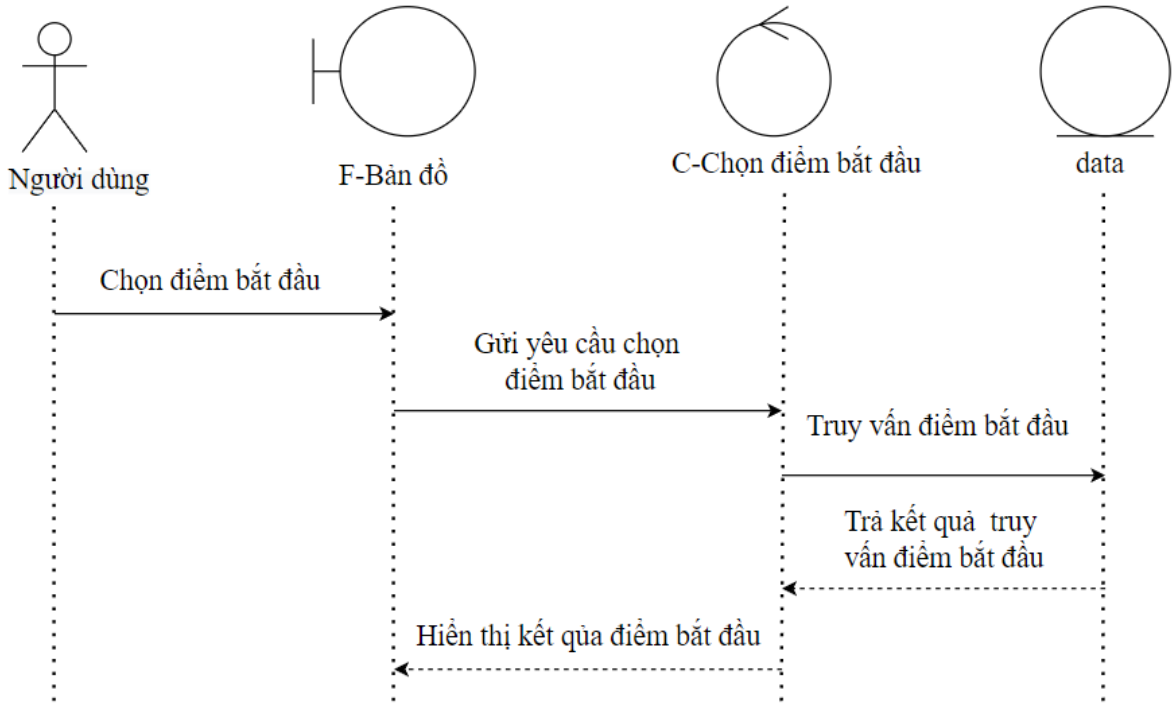


Hình 2.15: Biểu đồ Use Case “Tìm đường đi ngắn nhất”

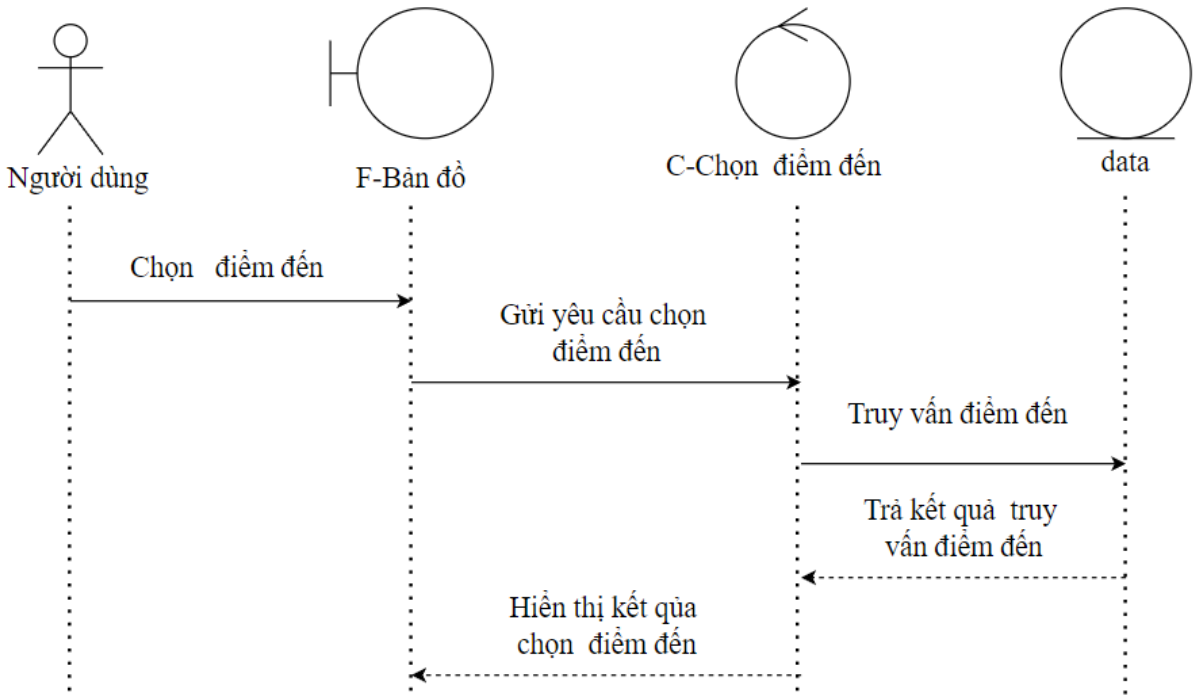
2.5. Biểu đồ tuần tự



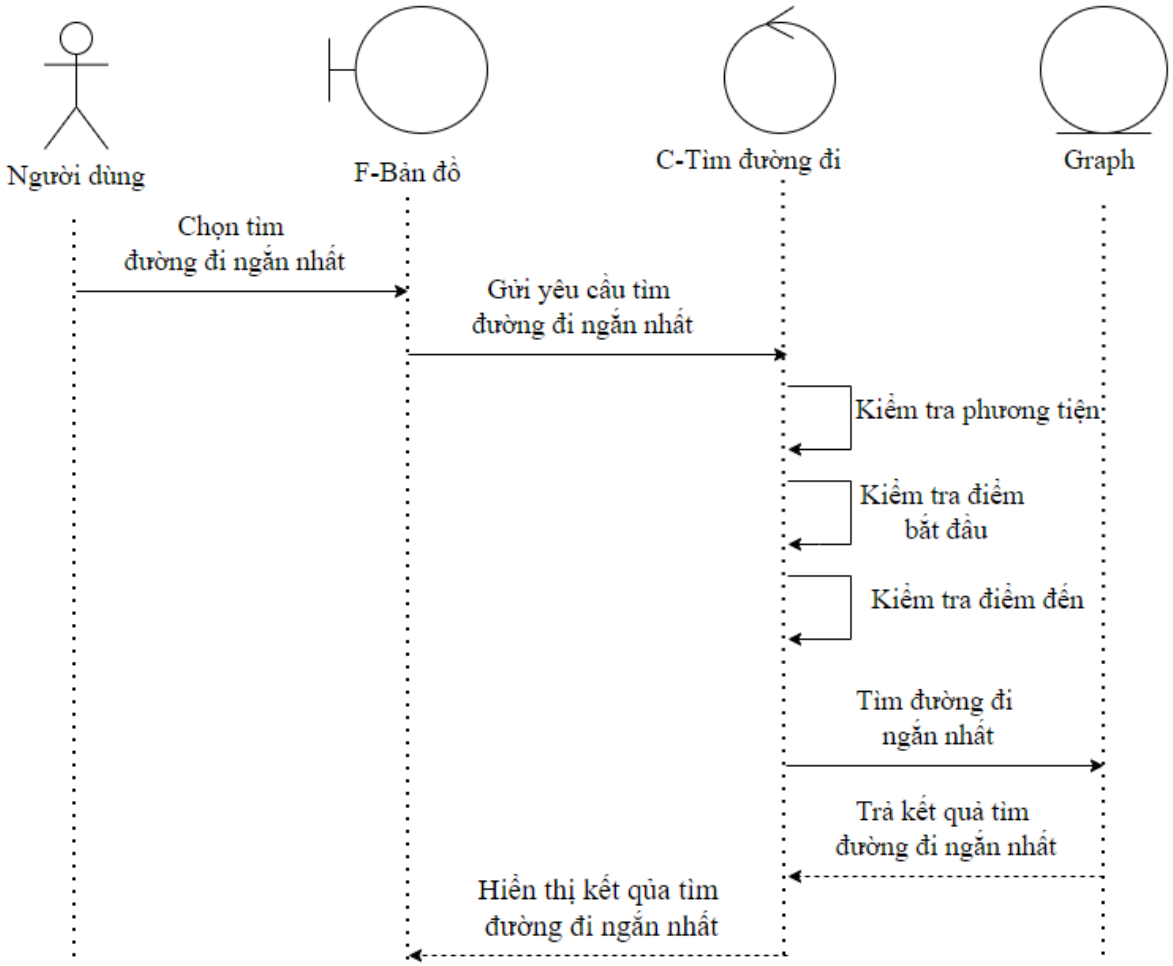
Hình 2.16: Biểu đồ Tuần tự "Chọn phương tiện"



Hình 2.17: Biểu đồ Tuần tự "Chọn điểm bắt đầu"

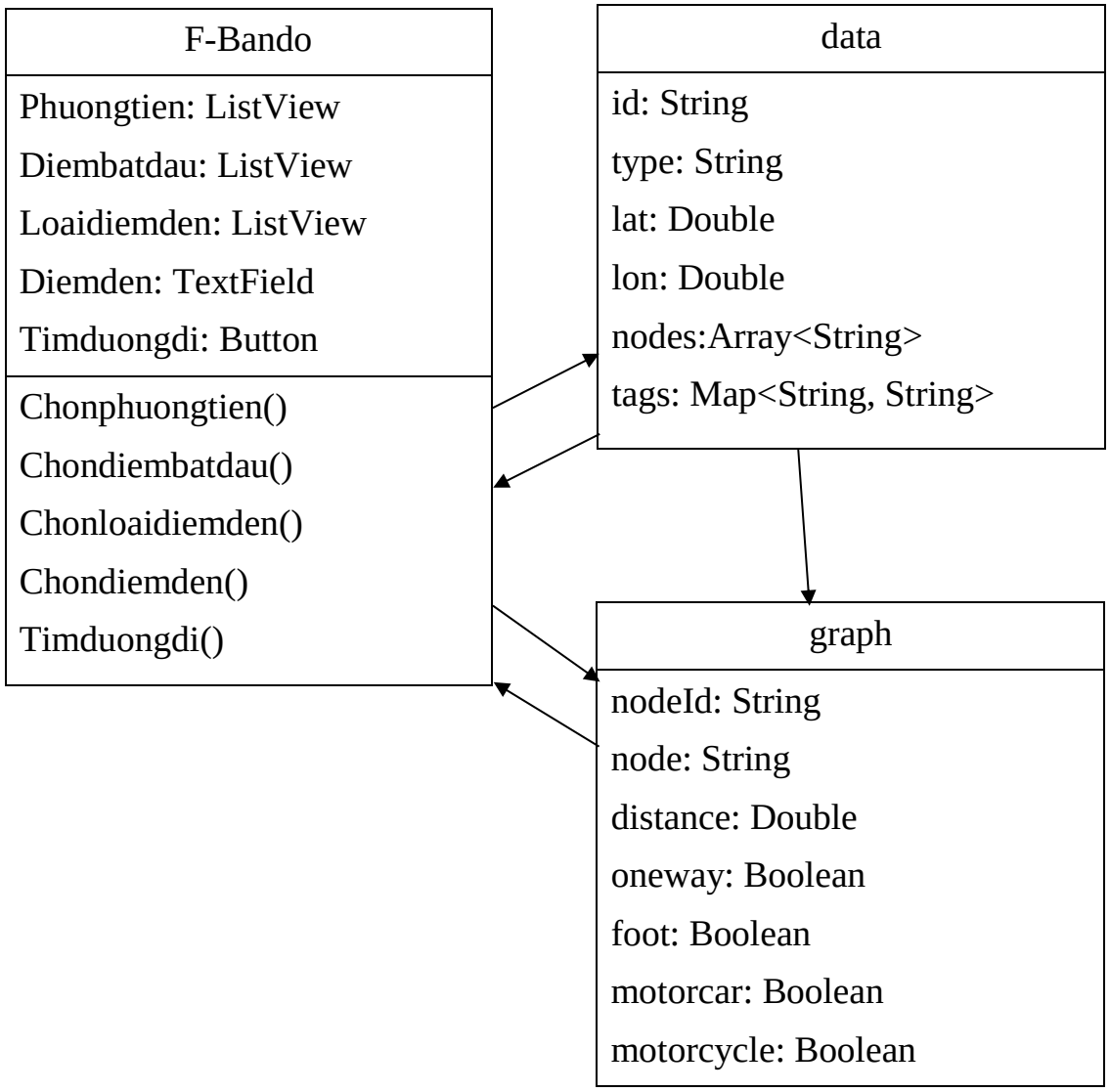


Hình 2.18: Biểu đồ Tuần tự “Chọn điểm đến”



Hình 2.19: Biểu đồ Tuần tự "Tìm đường đi ngắn nhất"

2.6. Biểu đồ lớp



Hình 2.20: Biểu đồ Lớp "Tìm đường đi ngắn nhất"

CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH VÀ THỬ NGHIỆM

3.1. Môi trường cài đặt

Chương trình được triển khai trên môi trường Node.js, một nền tảng phát triển cho ứng dụng web phía máy chủ. Node.js cho phép thực thi mã JavaScript trên máy chủ và làm cho JavaScript trở thành một ngôn ngữ lập trình đa mục đích, từ phía máy khách đến phía máy chủ. Máy chủ web server được xây dựng trên nền tảng Node.js, chạy ứng dụng và cung cấp dịch vụ web cho các yêu cầu từ các trình duyệt web.

Yêu cầu cấu hình máy tính

Yêu cầu phần cứng:

- ☞ Bộ xử lý (CPU): Bộ xử lý đủ mạnh để xử lý các yêu cầu của ứng dụng web. Một bộ xử lý đa nhân sẽ tối ưu hơn cho các ứng dụng đòi hỏi nhiều tác vụ đồng thời.
- ☞ Bộ nhớ (RAM): Tùy thuộc vào kích thước và tính năng của ứng dụng, khuyến nghị tối thiểu là 2GB RAM. Đối với các ứng dụng lớn hơn hoặc có lượng truy cập cao, cần có bộ nhớ lớn hơn.

Yêu cầu mạng:

- ☞ Kết nối Internet: Để tải về các gói phụ thuộc từ kho lưu trữ và sử dụng dịch vụ định vị vị trí hiện tại cũng như dịch vụ bản đồ, máy tính cần có kết nối Internet ổn định.

Yêu cầu phần mềm:

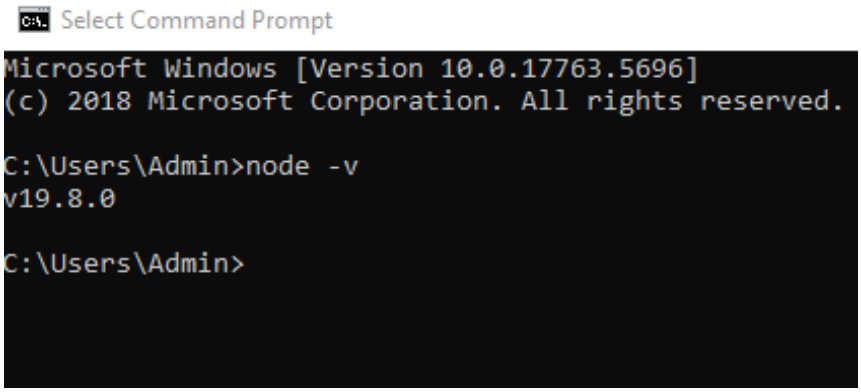
- ☞ Hệ điều hành (OS): Máy tính cần chạy một hệ điều hành tương thích với Node.js và các gói phụ thuộc của ứng dụng. Hầu hết các bản phân phối Linux (ví dụ: Ubuntu, CentOS) và phiên bản Windows hiện đại đều hỗ trợ.
- ☞ Node.js và npm: Cần cài đặt Node.js phiên bản Node.js cần phải tương thích với phiên bản mà ứng dụng yêu cầu.
- ☞ Trình duyệt web: Để kiểm tra và sử dụng ứng dụng, máy tính cần có một trình duyệt web như Chrome, Firefox, hoặc Edge.

  Cài nodejs: tải nodejs ở trang web https://nodejs.org/en/download

Sau khi tải về chạy file cài đặt.

Kiểm tra cài đặt bằng cách mở "cmd" (nhấn tổ hợp phím Window + R, gõ cmd, enter):

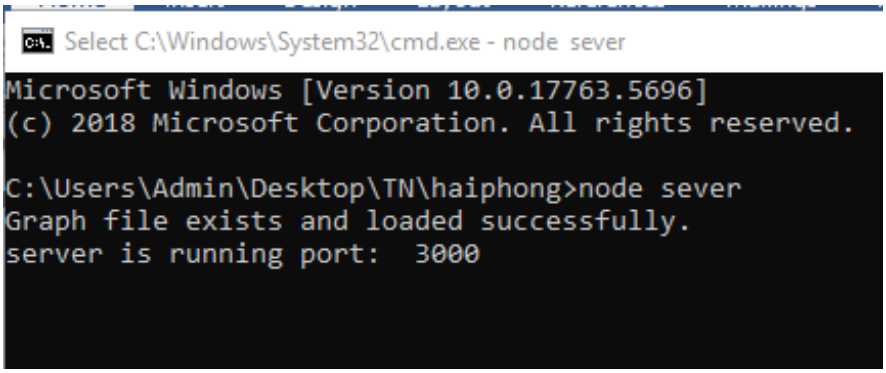
Gõ: node -v để kiểm tra



Hình 3.1: Kiểm tra Nodejs đã cài thành công

Di chuyển đến thư mục dự án và nhập lệnh: npm init --y để cài đặt các package có sẵn.

Sau đó chạy lênh: node sever.js để chạy sever

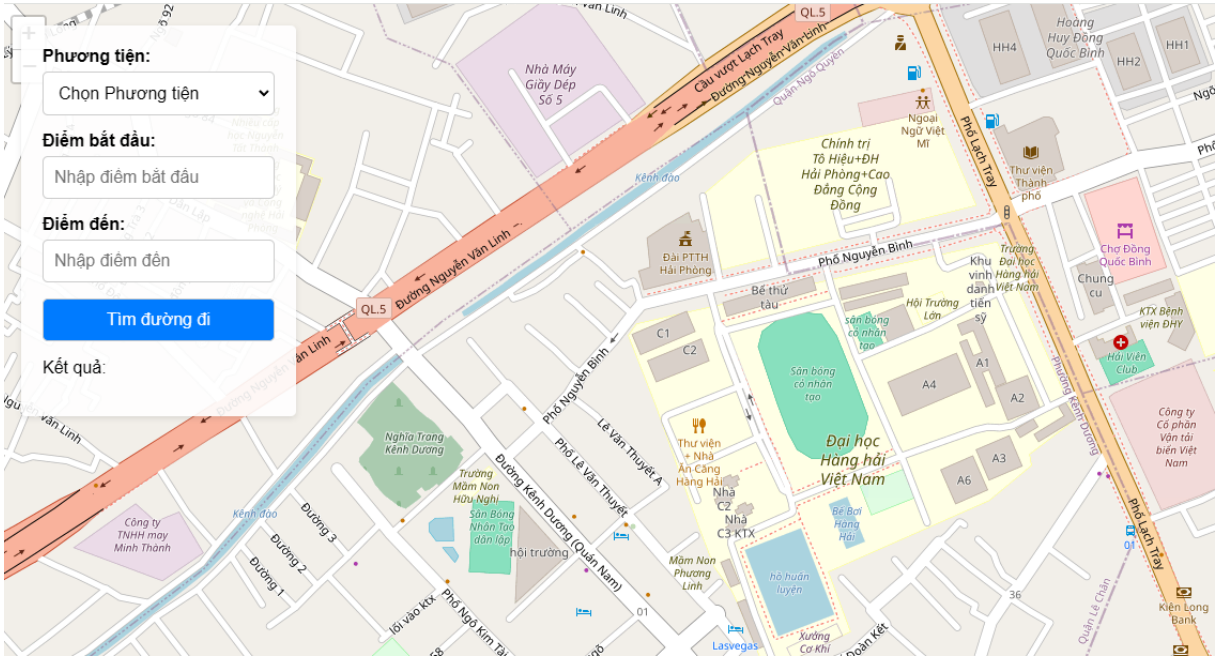


Hình 3.2: *Chạy sever*

Mở trình duyệt và gõ http://www.localhost:3000/

3.2. Giao diện chương trình

Trang web tìm đường ngắn nhất là một công cụ hữu ích giúp bạn dễ dàng lập kế hoạch và tìm đường đi ngắn nhất giữa hai điểm trên bản đồ. Với giao diện thân thiện và dễ sử dụng, bạn có thể chọn phương tiện di chuyển và điểm đến để nhận được lộ trình tối ưu về khoảng cách.



Hình 3.3: Giao diện website

Tính năng

- Chọn phương tiện di chuyển: Lựa chọn phương tiện di chuyển phù hợp với nhu cầu của bạn, bao gồm đi bộ, xe máy và ô tô.
- Chọn điểm bắt đầu: Nhập điểm bắt đầu hoặc nhấn trực tiếp trên bản đồ.
- Chọn điểm đến: Nhập điểm đến hoặc nhấn trực tiếp trên bản đồ.
- Tìm đường đi: Nhấn nút "Tìm đường đi" để nhận lộ trình chi tiết từ điểm bắt đầu đến điểm đến.

Hướng dẫn sử dụng

- Chọn phương tiện: Chọn loại phương tiện bạn muốn sử dụng để di chuyển (ô tô, xe máy hoặc đi bộ) thông qua danh sách có sẵn.
- Chọn điểm bắt đầu: Nhập điểm bắt đầu vào ô điểm bắt đầu hoặc nhấn trực tiếp vào vị trí mong muốn trên bản đồ.
- Chọn điểm đến: Nhập điểm đến vào ô điểm đến hoặc nhấn trực tiếp vào vị trí mong muốn trên bản đồ.
- Tìm đường đi: Nhấn nút "Tìm đường đi" và chờ đợi kết quả xuất hiện. Hệ thống sẽ hiển thị lộ trình từ điểm bắt đầu đến điểm đến.

Tuyến đường trên bản đồ

- Đường phổ thông (thường): Thường được biểu diễn bằng màu xám hoặc màu đen. Đây là các đoạn đường thông thường không có thuộc tính đặc biệt nào.

- Đường cao tốc: Thường được biểu diễn bằng màu xanh hoặc màu đỏ sáng. Đường cao tốc thường là các tuyến đường nhanh chóng và rộng lớn, có tốc độ cao.

- Đường phố: Có thể sử dụng màu xanh lá cây hoặc màu nâu. Đây là các đoạn đường thông thường trong các khu vực dân cư, thường có nhiều người đi lại.

- Đường đô thị: Thường được biểu diễn bằng màu hồng hoặc màu cam. Đường đô thị là các tuyến đường nằm trong khu vực thành thị, thường có đông đúc và nhiều giao thông.

- Đường cao cấp: Có thể được biểu diễn bằng màu vàng hoặc màu cam đậm. Đây là các đoạn đường chính trong thành phố hoặc khu vực có giá trị kinh tế cao, thường được bảo trì và sửa chữa đều đặn.

- Đường hỗn hợp: Có thể sử dụng màu trộn hoặc màu đặc biệt để biểu diễn các đoạn đường có thuộc tính đặc biệt không thuộc vào các loại trên.

*** Mục đích**

Trang web cung cấp một cách dễ dàng và thuận tiện để tìm kiếm và lập kế hoạch cho các chuyến đi của bạn qua các phương tiện phổ biến như ô tô, xe máy, đi bộ. Giúp người dùng tìm ra tuyến đường ngắn nhất giữa hai điểm, đồng thời cung cấp thông tin vị trí về các điểm đến phổ biến như bệnh viện, trạm xăng, nhà hàng và nhiều hơn nữa.

*** Mục Tiêu**

- Tiện lợi: Cung cấp một trải nghiệm dễ sử dụng và tiện lợi, giúp người dùng dễ dàng tìm kiếm và lập kế hoạch cho chuyến đi của mình.

- Nhanh chóng: Cung cấp thông tin và tìm kiếm tuyến đường nhanh chóng, giúp tiết kiệm thời gian cho người dùng.

- Chính xác: Cung cấp thông tin chính xác và chi tiết về các điểm đến, đảm bảo người dùng có thể chọn lựa đúng điểm đến cho nhu cầu của mình.

- Đa dạng: Cung cấp nhiều lựa chọn về phương tiện di chuyển và điểm đến, phù hợp với nhu cầu đa dạng của người dùng.

*** Ràng buộc**

❖ Dữ liệu đầu vào

- Phương tiện di chuyển: Người dùng có thể chọn phương tiện di chuyển bao gồm ô tô, xe máy hoặc đi bộ.

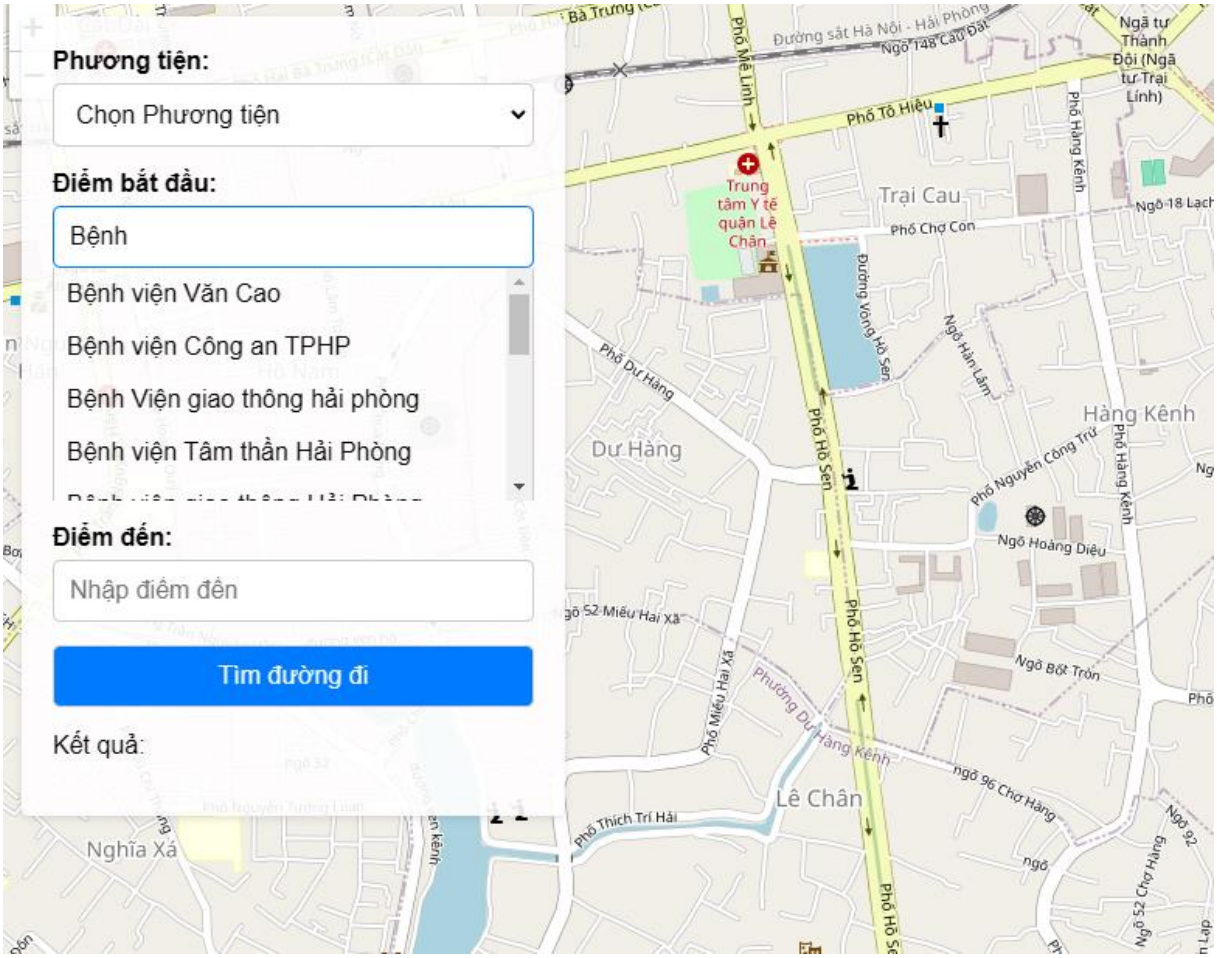
- Điểm bắt đầu: Người dùng có thể nhập điểm bắt đầu hoặc nhấn trực tiếp vào vị trí mong muốn trên bản đồ.
- Điểm đến: Người dùng có thể nhập điểm đến hoặc nhấn trực tiếp vào vị trí mong muốn trên bản đồ.

❖ Dữ liệu đầu ra

- Tuyến đường: Hệ thống sẽ cung cấp một tuyến đường ngắn nhất bắt đầu và điểm đến, dựa trên phương tiện di chuyển điểm bắt đầu và điểm đến được chọn.
- Hiển thị đường đi trên bản đồ: Đường đi ngắn nhất sẽ được tô màu trên bản đồ từ điểm bắt đầu và điểm đến.

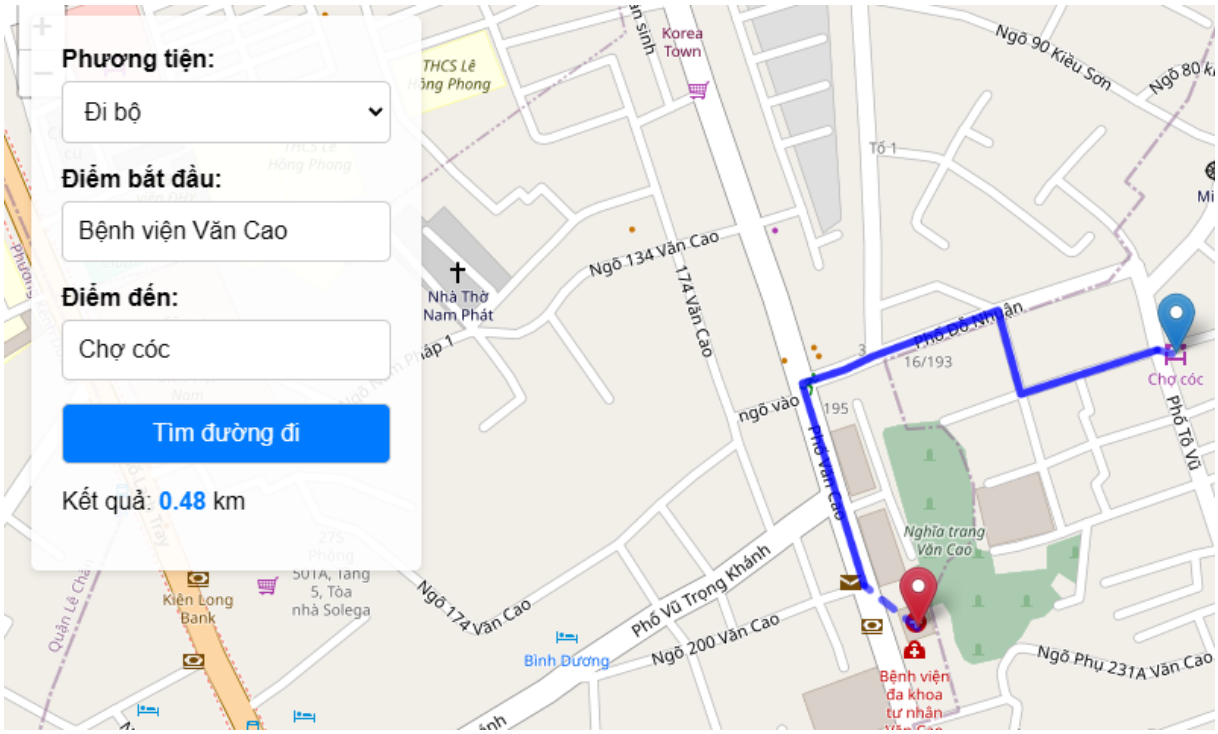
3.3. Thử nghiệm

3.3.1. Thử nghiệm gợi ý tìm kiếm địa điểm



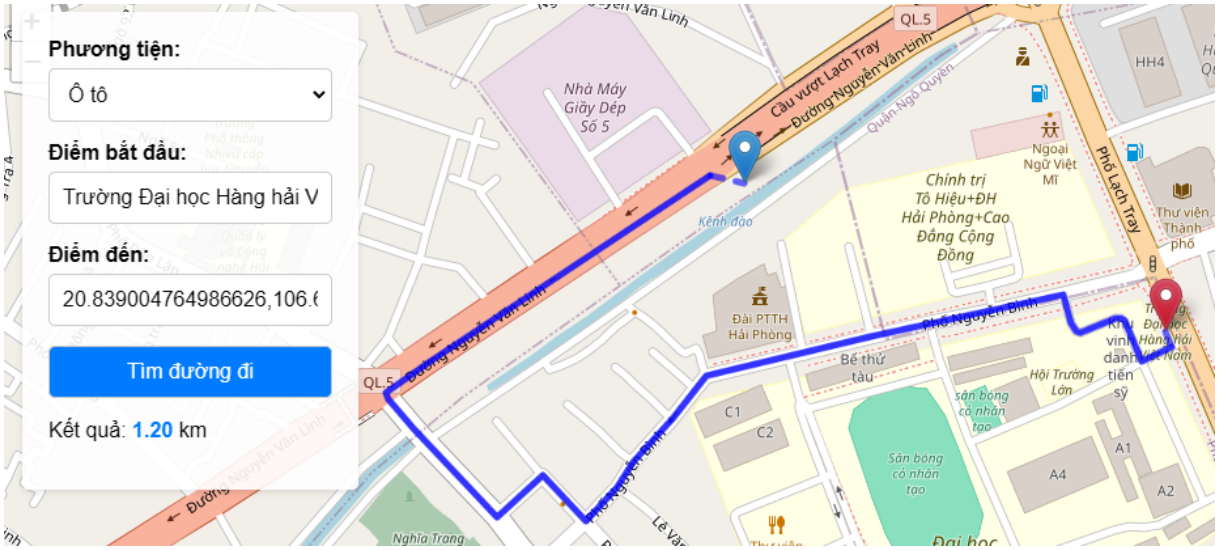
Hình 3.4: Kết quả thử nghiệm gợi ý tìm kiếm địa điểm

3.3.2. Trường hợp đường đi không có đường một chiều

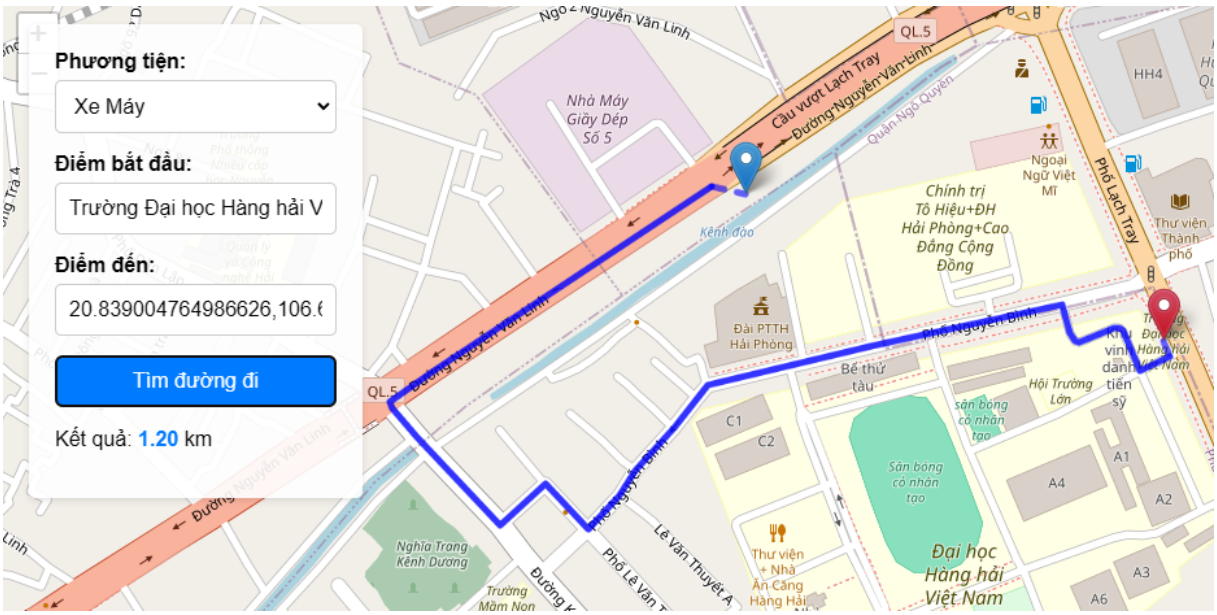


Hình 3.5: *Kết quả chạy thử đường đi không có đường một chiều*

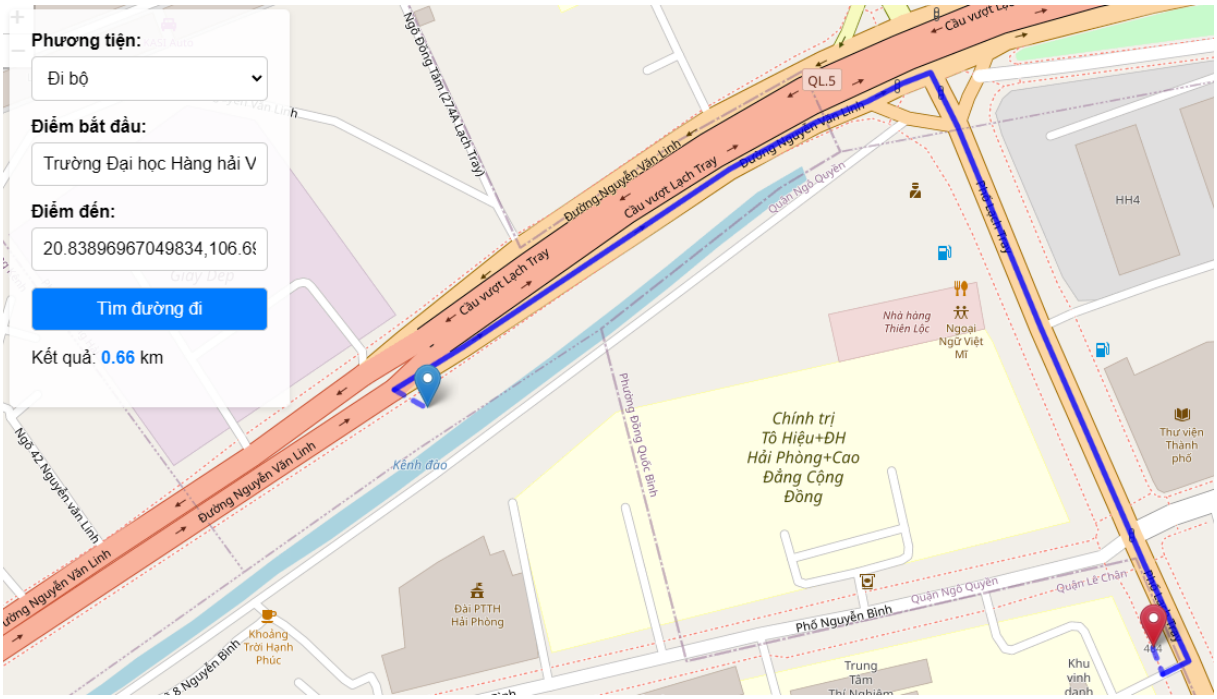
3.3.3. Trường hợp đường đi có đường một chiều



Hình 3.6: Kết quả chạy thử đi vào đường một chiều với ô tô

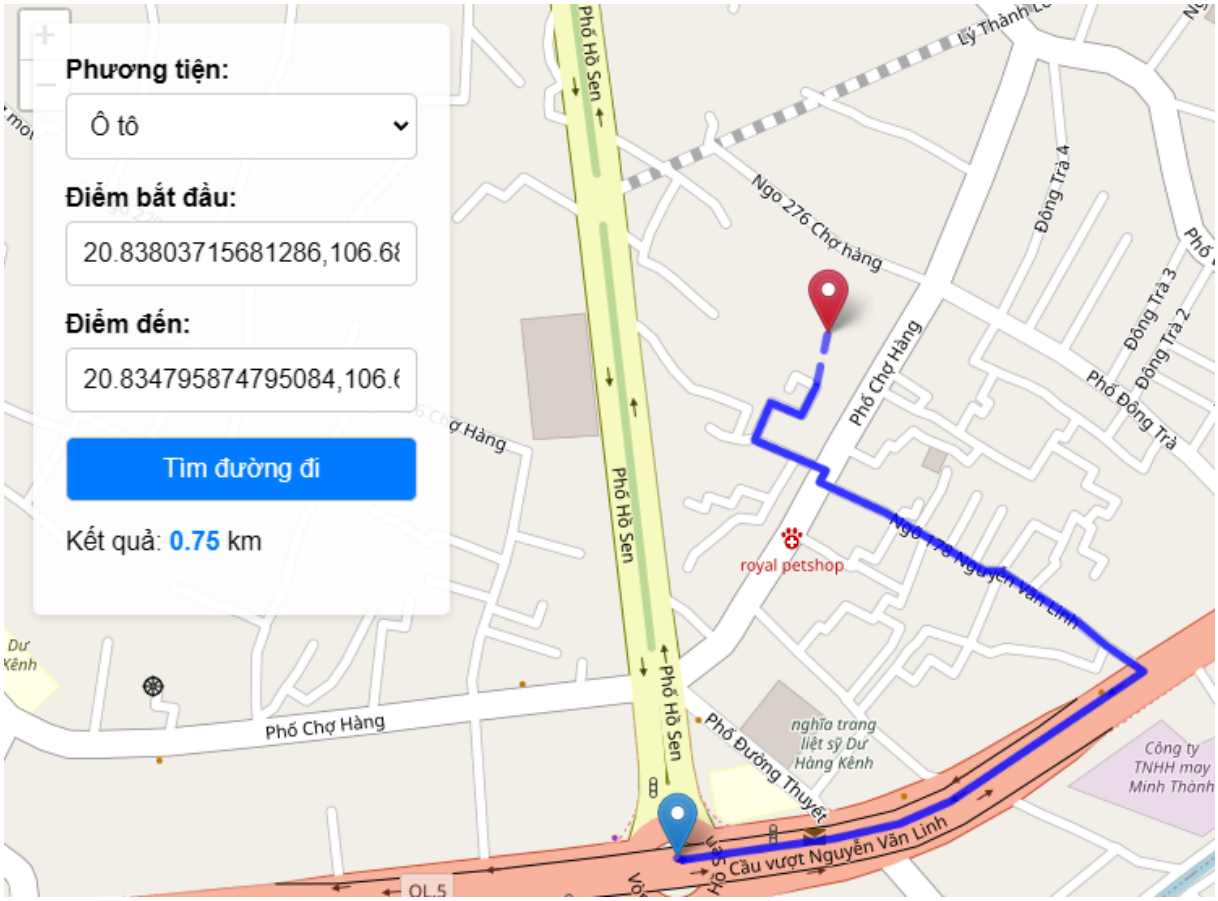


Hình 3.7: Kết quả chạy thử đi vào đường một chiều với xe máy

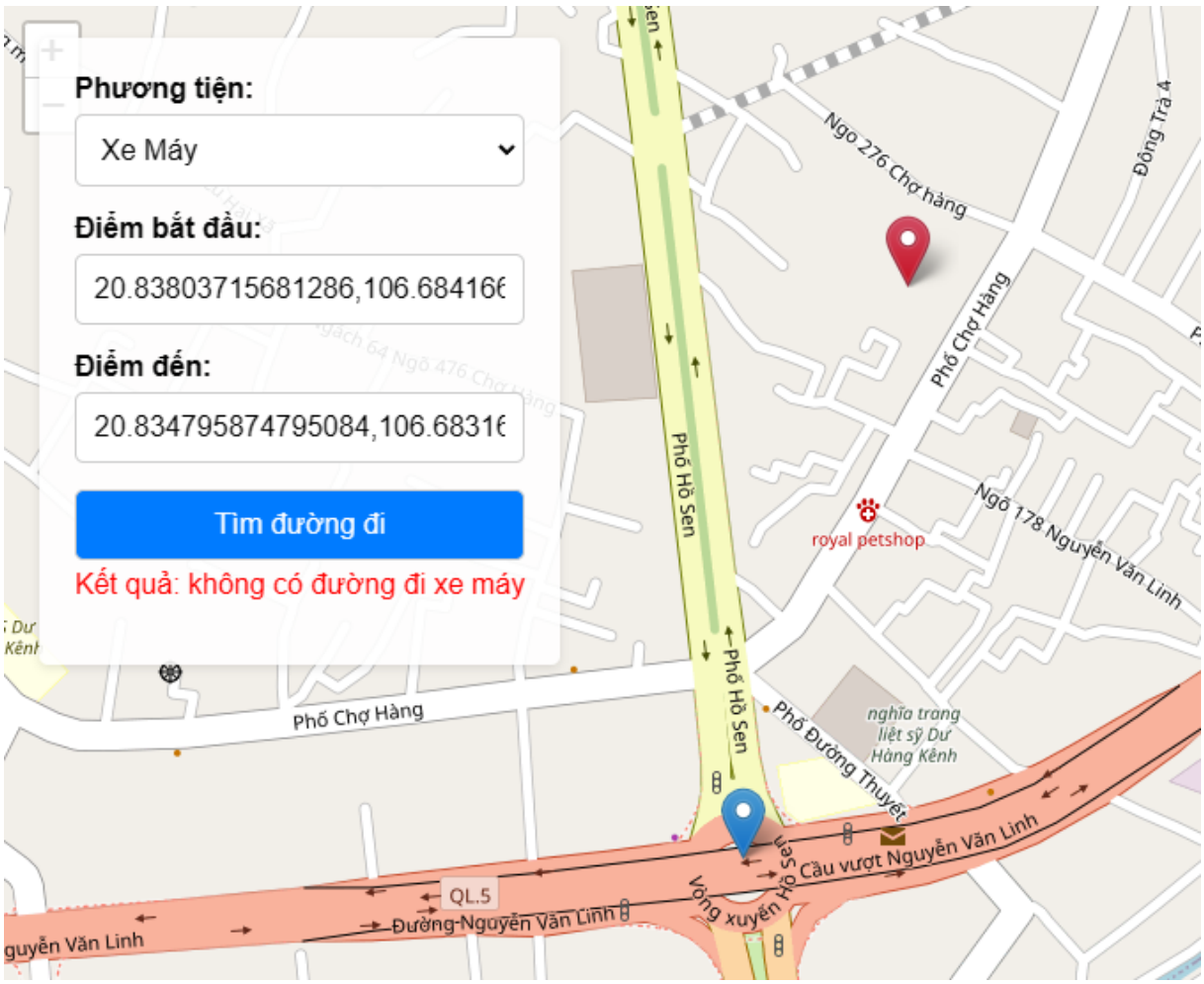


Hình 3.8: Kết quả chạy thử đi vào đường một chiều với đi bộ

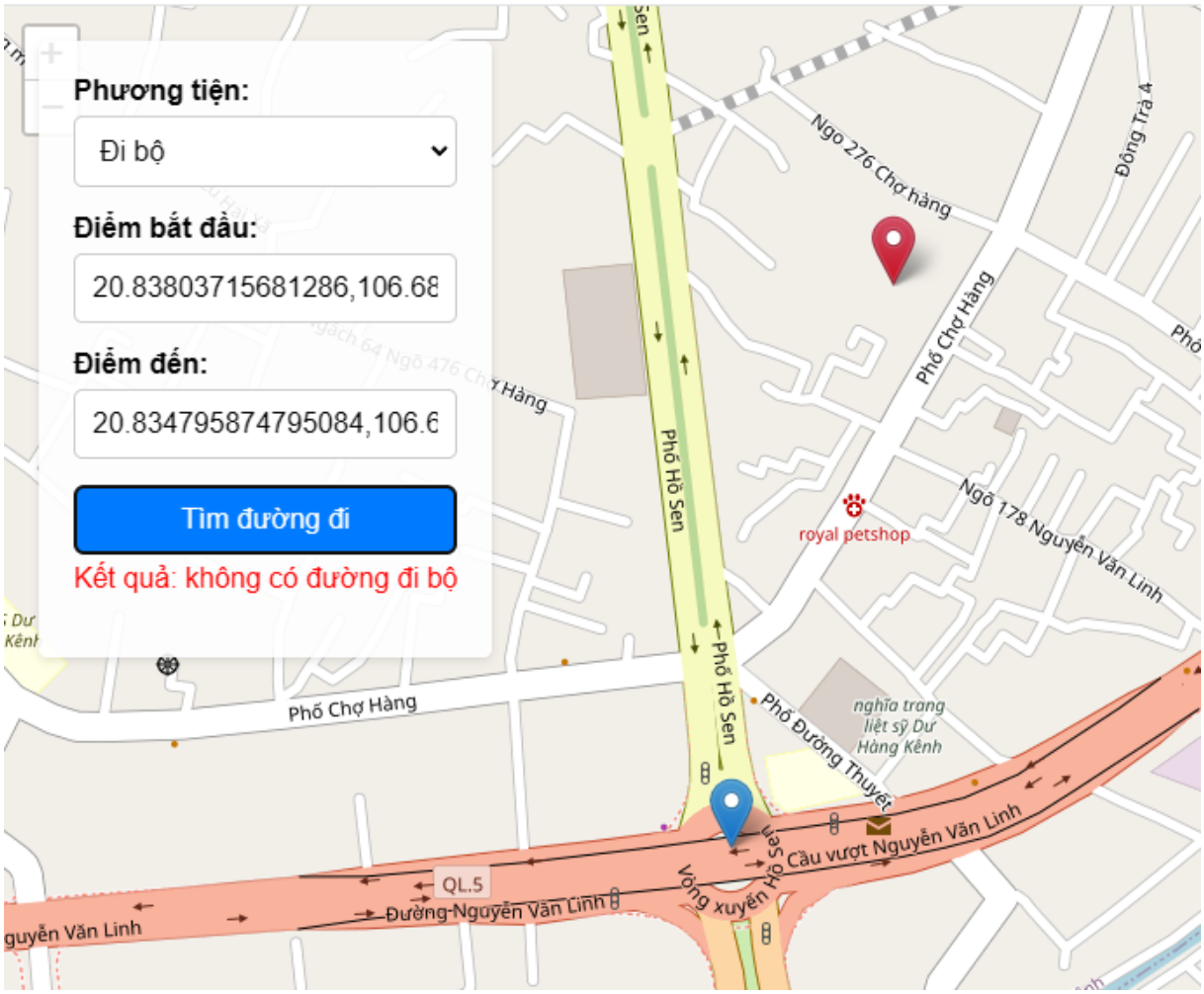
3.3.4. Trường hợp đường đi cấm đi bộ và xe máy



Hình 3.9: Kết quả chạy thử đi vào đường cấm đi bộ và xe máy với ô tô

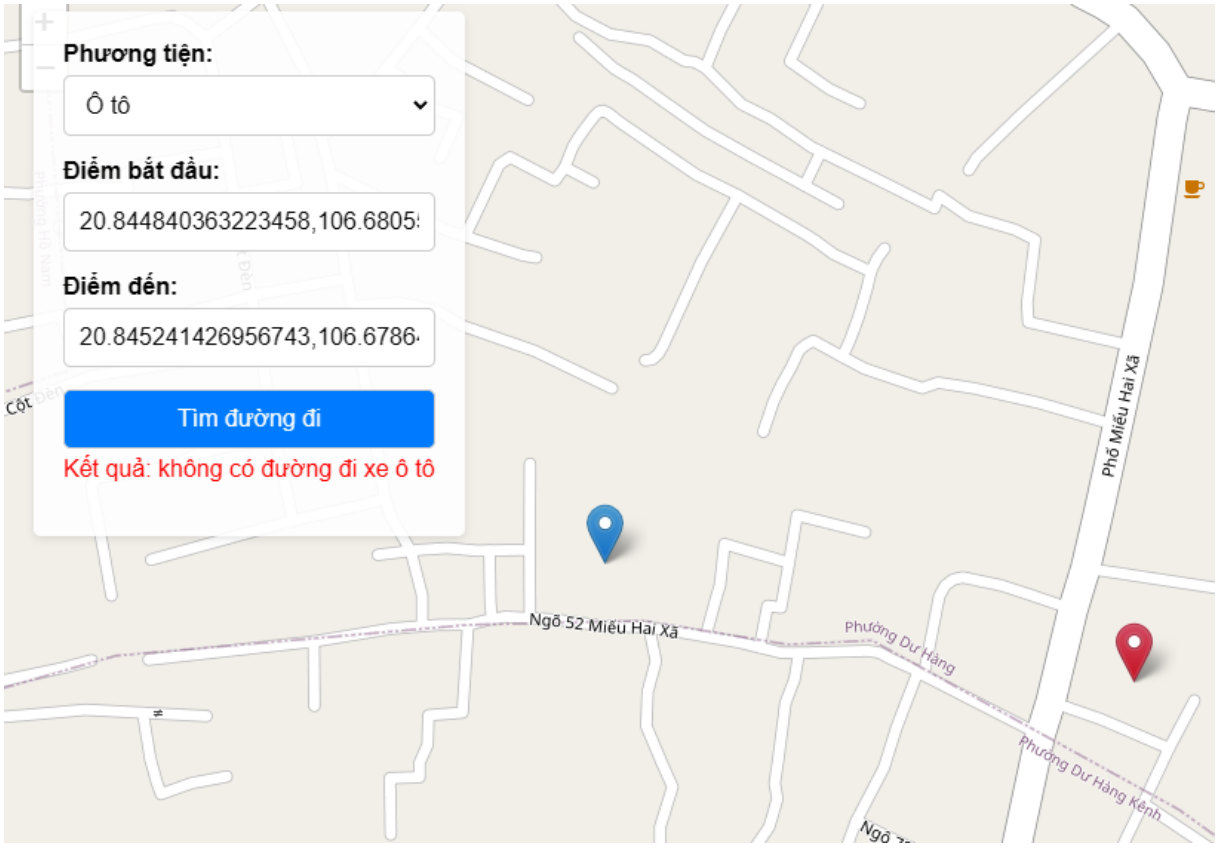


Hình 3.10: Kết quả chạy thử đi vào đường cấm đi bộ và xe máy với xe máy

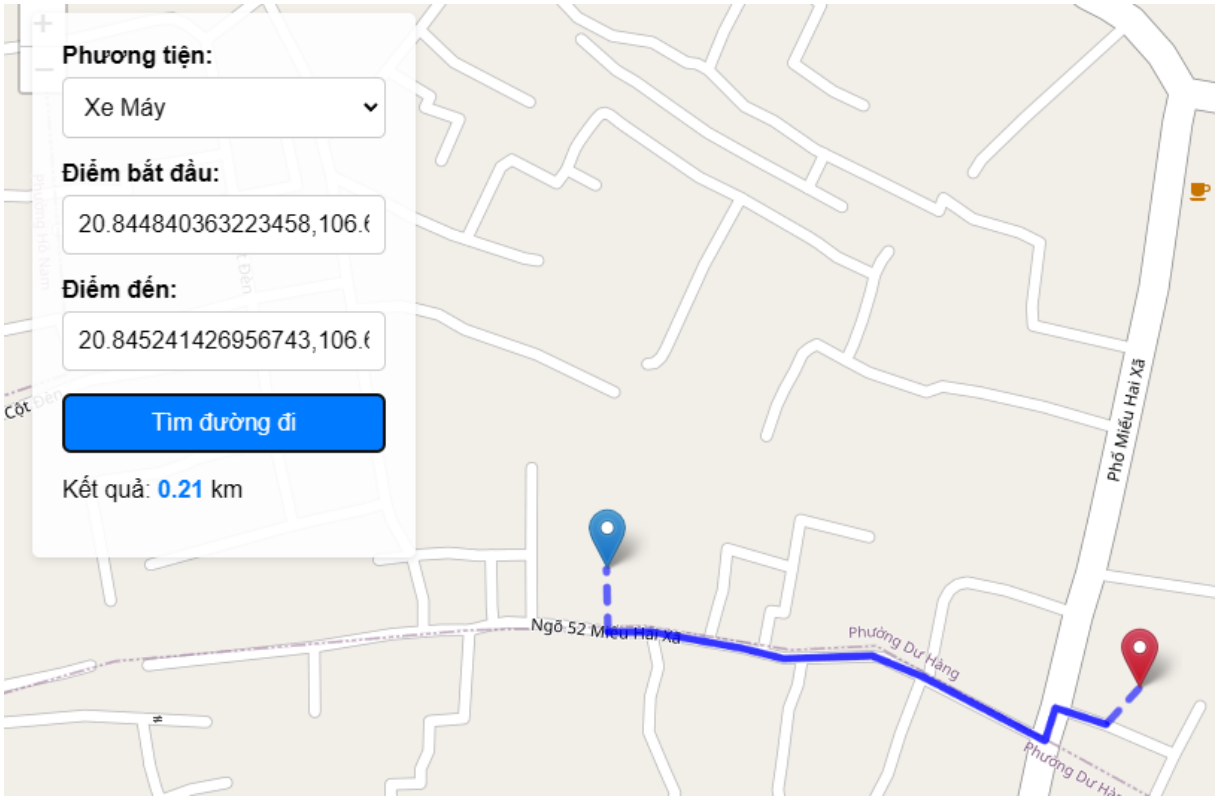


Hình 3.11: Kết quả chạy thử đi vào đường cấm đi bộ và xe máy với đi bộ

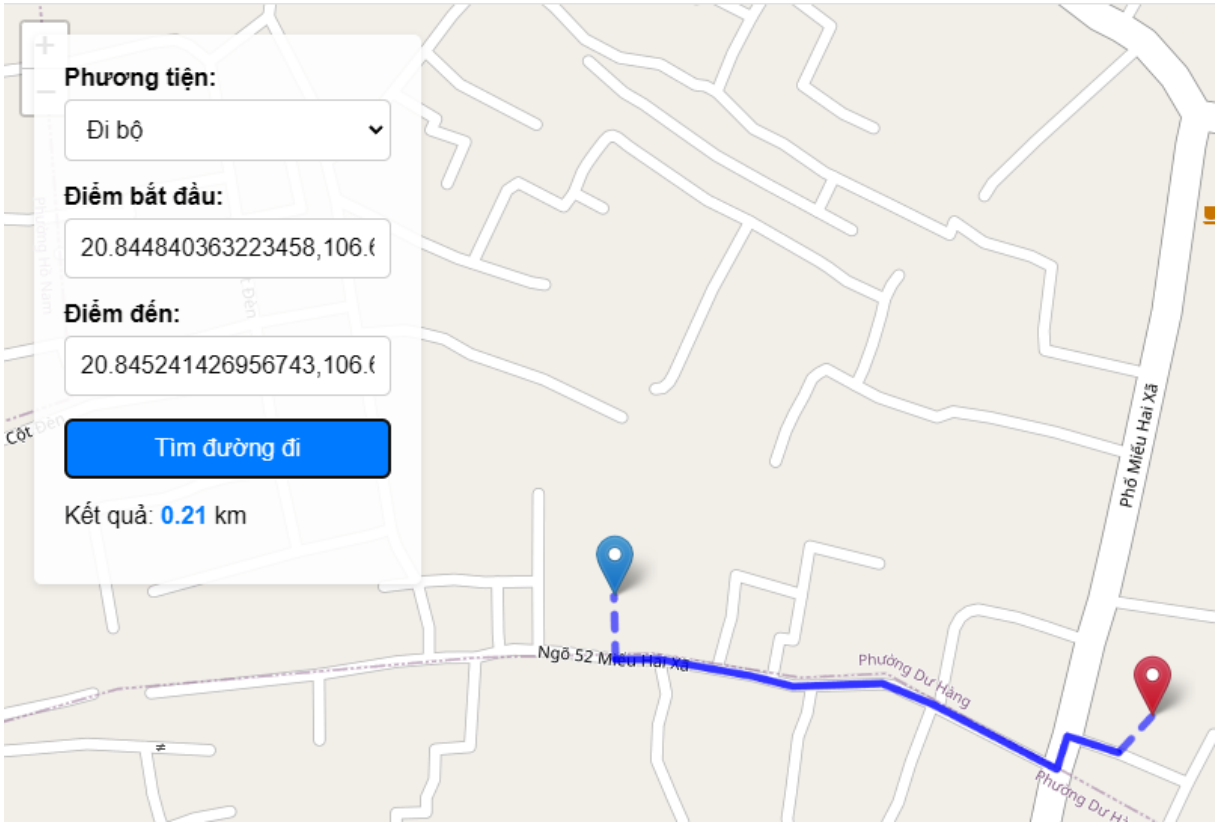
3.3.5. Trường hợp đường đi cấm xe ô tô



Hình 3.12: Kết quả chạy thử đi vào đường cấm đi ô tô với xe ô tô



Hình 3.13: Kết quả chạy thử đi vào đường cấm đi ô tô với xe máy



Hình 3.14: Kết quả chạy thử đi vào đường cấm đi ô tô với đi bộ

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- **Kết quả đạt được**

Mục tiêu của hệ thống là cung cấp cho người dùng một giao diện đơn giản và thuận tiện để tìm đường đi ngắn nhất giữa hai điểm trên bản đồ, dựa trên các yếu tố như loại phương tiện di chuyển, điểm bắt đầu và điểm đến. Sau một thời gian tìm hiểu và nghiên cứu đề tài này, em đã đạt được một số kết quả như sau:

- Tìm đường đi ngắn nhất: Hệ thống sẽ cung cấp một đường đi ngắn nhất từ điểm bắt đầu đến điểm đích dựa trên thông tin về phương tiện di chuyển, điểm bắt đầu và điểm đích đã được người dùng chọn.
- Thông tin địa điểm: Ngoài đường đi ngắn nhất hệ thông cũng hiển thị các loại điểm bắt đầu hoặc điểm đến mà người dùng không biết rõ tên của điểm đến bằng cách gợi ý tìm kiếm địa điểm từ đó có thể lựa chọn đường đi ngắn nhất đến điểm được lựa chọn. Hoặc người dùng có thể nhấp trực tiếp vào bản đồ với vị trí mong muốn là điểm bắt đầu hoặc điểm đến.
- Giao diện thân thiện với người dùng: Giao diện của hệ thống được thiết kế đơn giản và dễ sử dụng, giúp người dùng dễ dàng nhập thông tin và nhận kết quả một cách nhanh chóng.
- Hiệu suất cao: Hệ thống được xây dựng với một kiến trúc hiệu quả, đảm bảo việc tìm kiếm đường đi diễn ra nhanh chóng.
- Dữ liệu bản đồ: hiểu thêm về cách truy vấn về phạm vi, dữ liệu cần thiết tải và lưu trữ dữ liệu bản đồ từ OSM để triển khai dự án website tìm đường đi ngắn nhất.
- Xử lý dữ liệu đồ thị: hiểu về cấu trúc thông tin dữ liệu bản đồ được cấu tạo như nào, các thông tin, kiểu dữ liệu được biểu diễn. Từ đó có thể phân tích, sử dụng để tạo, lưu trữ và sử dụng dữ liệu đồ thị graph từ file dữ liệu được tải về ban đầu.
- Áp dụng thuật toán Dijkstra: áp dụng thành công thuật toán Dijkstra tìm đường đi giữa hai điểm, phương tiện di chuyển được chọn với đồ thị graph có chứa đầy đủ thông tin để triển khai thuật toán Dijkstra như: điểm và các điểm kề với nó với khoảng cách từ điểm đó tới các điểm kề, và các thông tin khác như đường một chiều, cấm ô tô, cấm xe máy, cấm đi bộ. Từ đó có thể triển khai và điều chỉnh thuật toán Dijkstra khi có thêm thông tin phương tiện đi lại như ô tô, xe máy hoặc đi bộ.

- Hiểu thêm về cách tính khoảng cách giữa hai điểm thông qua tọa độ của chúng áp dụng công thức Haversine.

- Các kiến thức về nodejs express, html, css, js và thư viện leaflet giúp hiển thị bản đồ. Được áp dụng các kiến thức đã học trong trường và kinh nghiệm trong quá trình thực hiện dự án.

- **Hạn chế**

- Phạm vi dữ liệu: Chương trình không xử lý được với dụng lượng dữ liệu lớn, dữ liệu tải xuống còn chưa đầy đủ.

- Hiệu suất máy tính không đủ: Xử lý dữ liệu lớn và tính toán phức tạp của thuật toán Dijkstra có thể đòi hỏi sự hiệu suất cao từ máy tính. Máy tính không đủ mạnh có thể dẫn đến việc giảm hiệu suất hoặc thậm chí làm gián đoạn quá trình tính toán.

- Bảo mật dữ liệu: Chưa có phương pháp tốt để bảo mật dữ liệu bản đồ.

- Chỉ cho phép người dùng chọn những loại điểm đến có sẵn hoặc điểm trên phạm vi nhỏ của thành phố Hải Phòng, và với ba loại phương tiện di chuyển là ô tô, xe máy, hoặc đi bộ. Chưa tính toán được các chi tiết quan trọng như thời gian, chi phí, địa hình, tình trạng giao thông các tuyến đường.

- Tính chính xác công thức Haversine: Công thức Haversine thiếu chính xác với các điểm nằm trên đường cong như đường đồi núi hay đường thung lũng.

- Vị trí hiện tại: Chưa tích hợp được chức năng hiển thị vị trí hiện tại của người dùng.

- **Hướng phát triển**

- Tối ưu hóa thuật toán tìm đường đi: Nghiên cứu và triển khai các thuật toán tìm đường đi mới hoặc cải tiến thuật toán Dijkstra để xử lý các bản đồ lớn và phức tạp một cách hiệu quả hơn.

- Xử lý dữ liệu lớn: Nghiên cứu và tìm hiểu các phương pháp lưu trữ, sử dụng với dữ liệu lớn tốt hơn.

- Định vị vị trí và bảo mật: Tìm hiểu nâng cao độ chính xác và bảo mật vị trí người dùng.

- Tích hợp các tính năng mới: Mở rộng hệ thống bằng cách thêm tính năng hỗ trợ các phương tiện di chuyển khác nhau như: xe buýt, xe điện, hoặc các phương tiện khác. Cung cấp thông tin về các địa điểm quan trọng như trạm

xe buýt, trạm sạc điện, cửa hàng … các thông tin quan trọng khác của địa điểm như giờ mở cửa, lịch làm việc, hình ảnh, đánh giá của người dùng.

Mặc dù đã rất cố gắng trong việc nghiên cứu và thực hiện đồ án, nhưng do thời gian và kiến thức, tài nguyên máy tính của em còn hạn chế nên chương trình, đồ án chỉ dừng lại ở mức ứng dụng cơ bản trong một phạm vi địa lý hạn chế. Để triển khai được ứng dụng vào thực tế cần thêm rất nhiều kiến thức, hiểu biết về các thuật toán trong lĩnh vực học sâu cũng như hiểu biết ở các lĩnh vực công nghệ nâng cao khác nữa. Do đó, đồ án cũng sẽ không tránh khỏi những thiếu sót. Em rất mong nhận được ý kiến góp ý từ các thầy cô cũng như các bạn để hoàn thiện đồ án được tốt hơn.

TÀI LIỆU THAM KHẢO

[1]. Robbins, Jennifer. Learning Web Design. 5th ed., O'Reilly Media, Inc., 2018.

[2]. Đoàn Thanh Nghị, Giáo trình Lập trình Web, Đại học Quốc gia TP.HCM, 2022.

[3]. Trương Ninh Thuận, Đặng Đức Hạnh, Giáo trình Phân tích thiết kế hướng đối tượng, Đại học Quốc gia Hà Nội, 2016.

[4]. Nguyễn Tuệ, Giáo trình nhập môn hệ cơ sở dữ liệu, Nhà xuất bản giáo dục (tái bản lần thứ nhất), 2009.

[5]. Dijkstra, E. W. "A Note on Two Problems in Connexion with Graphs." Numerische Mathematik, vol. 1, no. 1, pp. 269-271. Edited by A. S. Householder, R. Sauer, E. Stiefel, and A. Walther, Springer, 1959.

[6]. Google. Google Maps. Google, n.d. Web. https://www.google.com/maps

[7]. Bing Map. Microsoft, n.d. Web. https://www.bing.com/maps.

[8]. OpenStreetMap, OpenStreetMap Foundation, n.d. Web. https://openstreetmap.org

[9]. Node.js. Node.js Foundation, n.d. Web. https://nodejs.org

[10]. Express.js. Express.js Contributors, n.d. Web. Accessed on multiple dates. Last accessed 22 May 2024. https://expressjs.com

[11]. Overpass Turbo. n.d. Web. https://overpass-turbo.eu

[12]. Leaflet.js. Leaflet Contributors, n.d. Web. https://leafletjs.com/.

[13]. "Haversine Formula." Wikipedia, Wikimedia Foundation, 15 May 2024. Web. https://en.wikipedia.org/wiki/Haversine_formula