

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



THỰC TẬP CƠ SỞ

ĐỀ TÀI:
XÂY DỰNG TRANG WEB XE BUÝT

Giảng viên hướng dẫn : TS. Đào Ngọc Phong

Sinh viên thực hiện : Nguyễn Tuấn Minh

Mã sinh viên : B22DCKH077

Hà Nội, năm 2025

1.1 Tổng quan hệ thống

1.1.1 Giới thiệu đề tài

Trong bối cảnh đô thị hóa ngày càng phát triển, giao thông công cộng đóng vai trò quan trọng trong việc giảm thiểu ùn tắc giao thông và ô nhiễm môi trường. Xe buýt, với khả năng vận chuyển đồng thời nhiều hành khách, là một trong những phương tiện giao thông công cộng phổ biến và hiệu quả nhất. Tuy nhiên, việc sử dụng xe buýt vẫn còn nhiều rào cản, trong đó có việc thiếu thông tin chính xác về lộ trình, thời gian chờ đợi, và vị trí hiện tại của xe.

Đề tài "Xây dựng trang web xe buýt" được thực hiện nhằm tạo ra một nền tảng trực tuyến cung cấp thông tin đầy đủ và chính xác về hệ thống xe buýt, giúp người dùng dễ dàng tìm kiếm thông tin về các tuyến xe, theo dõi vị trí xe buýt theo thời gian thực, và tìm đường đi giữa hai điểm bất kỳ. Hệ thống được xây dựng sử dụng các công nghệ hiện đại như ReactJS cho frontend, Express.js cho backend, SQL Server cho cơ sở dữ liệu, và WebSocket để theo dõi vị trí xe buýt theo thời gian thực.

Với sự phát triển của công nghệ thông tin và Internet, việc cung cấp thông tin giao thông công cộng trực tuyến đã mở ra cơ hội lớn để cải thiện trải nghiệm người dùng và khuyến khích việc sử dụng phương tiện công cộng. Tuy nhiên, việc tạo ra một hệ thống thông tin xe buýt hiệu quả đòi hỏi các tính năng đặc thù, bao gồm hiển thị bản đồ tương tác, cập nhật vị trí xe buýt theo thời gian thực, và thuật toán tìm đường.

Đề tài này được thực hiện trong bối cảnh đó, nhằm cung cấp một giải pháp giúp người dùng dễ dàng tiếp cận và sử dụng dịch vụ xe buýt thông qua một nền tảng trực tuyến dễ sử dụng, hiệu quả và chính xác. Hệ thống không chỉ hỗ trợ người dùng tìm kiếm thông tin về các tuyến xe buýt mà còn giúp họ lập kế hoạch di chuyển, theo dõi vị trí xe buýt theo thời gian thực, và nhận thông báo về các thay đổi lịch trình hoặc sự cố.

Với việc ứng dụng các công nghệ hiện đại, hệ thống này sẽ đáp ứng được nhu cầu về giao diện thân thiện, hiệu suất cao và khả năng mở rộng trong tương lai. Điều này đặc biệt quan trọng trong bối cảnh thành phố thông minh đang ngày càng phát triển, yêu cầu các hệ thống thông tin giao thông phải linh hoạt và có khả năng xử lý dữ liệu lớn.

Với xu hướng số hóa và hóa giao thông đô thị, việc phát triển một nền tảng thông tin xe buýt không chỉ giúp cải thiện trải nghiệm người dùng mà còn góp phần vào việc xây dựng một hệ thống giao thông thông minh và bền vững, từ đó nâng cao chất lượng cuộc sống đô thị.

1.1.2 Lý do chọn đề tài

Đề tài "Xây dựng trang web xe buýt" được chúng em lựa chọn làm đề tài nhằm nghiên cứu và phát triển một nền tảng thông tin trực tuyến chuyên biệt dành cho lĩnh vực giao thông công cộng, cụ thể là xe buýt. Hệ thống giao thông công cộng hiện nay, đặc biệt là xe buýt, đang ngày càng được chú trọng phát triển để giảm thiểu ùn tắc giao thông và ô nhiễm môi trường. Tuy nhiên, việc tiếp cận thông tin về các tuyến xe buýt, lịch trình, và vị trí hiện tại của xe vẫn còn nhiều hạn chế, gây khó khăn cho người dùng. Do đó, việc xây dựng một

trang web thông tin xe buýt giúp người dùng dễ dàng tiếp cận và sử dụng dịch vụ xe buýt là một giải pháp thiết thực và cần thiết.

Qua đề tài này, chúng em không chỉ mong muốn tạo ra một sản phẩm có tính ứng dụng cao mà còn có cơ hội để phát triển và hoàn thiện các kỹ năng lập trình web, đồng thời áp dụng các công nghệ hiện đại trong lĩnh vực phát triển phần mềm. Việc sử dụng các công nghệ như ReactJS cho frontend và Express.js cho backend sẽ giúp chúng em hiểu rõ hơn về cách kết hợp các công nghệ frontend mạnh mẽ với hệ thống backend vững chắc để tạo ra một ứng dụng web đầy đủ tính năng và hiệu quả. ReactJS mang lại cho người dùng trải nghiệm mượt mà, linh hoạt, đồng thời đảm bảo khả năng mở rộng khi ứng dụng phát triển. Express.js, với khả năng xử lý các tác vụ phức tạp, bảo mật và dễ dàng tích hợp với các hệ thống khác, sẽ là nền tảng lý tưởng để phát triển một hệ thống backend ổn định và mạnh mẽ.

Bên cạnh việc học hỏi và áp dụng những kiến thức chuyên môn, đề tài này còn giúp chúng em rèn luyện kỹ năng giải quyết vấn đề thực tế, từ việc thiết kế kiến trúc hệ thống, xây dựng các tính năng đến hóa hiệu suất và bảo mật cho ứng dụng. Đặc biệt, chúng em sẽ tập trung vào việc phát triển các tính năng quan trọng của một trang web thông tin xe buýt như hiển thị bản đồ tương tác, theo dõi vị trí xe buýt theo thời gian thực, tìm đường đi, và cung cấp thông tin chi tiết về các tuyến xe buýt.

Một lý do quan trọng khác là tính thực tiễn và tác động xã hội của đề tài. Trong bối cảnh ô nhiễm môi trường và ùn tắc giao thông đang là vấn đề nghiêm trọng tại các đô thị lớn, việc khuyến khích sử dụng phương tiện giao thông công cộng, đặc biệt là xe buýt, là một giải pháp hiệu quả. Bằng cách cung cấp một nền tảng thông tin xe buýt dễ sử dụng và chính xác, đề tài của chúng em góp phần vào việc thúc đẩy sử dụng phương tiện công cộng, từ đó giảm thiểu ô nhiễm môi trường và ùn tắc giao thông.

Bằng việc thực hiện đề tài này, chúng em mong muốn có thể tạo ra một sản phẩm có giá trị thực tiễn, giúp người dùng dễ dàng tiếp cận và sử dụng dịch vụ xe buýt, đồng thời áp dụng những kỹ thuật, công cụ và phương pháp hiện đại trong việc xây dựng một ứng dụng web hoàn chỉnh. Đây cũng là một bước chuẩn bị quan trọng để chúng em phát triển sự nghiệp trong lĩnh vực công nghệ thông tin, khi mà các ứng dụng thông minh trong lĩnh vực giao thông đang ngày càng phát triển mạnh mẽ và đóng vai trò quan trọng trong việc xây dựng thành phố thông minh.

1.1.3 Mục tiêu

Mục tiêu chính của đề tài "Xây dựng trang web xe buýt" là tạo ra một nền tảng thông tin toàn diện và dễ sử dụng về hệ thống xe buýt, nhằm cải thiện trải nghiệm của người dùng và khuyến khích việc sử dụng phương tiện giao thông công cộng. Cụ thể, đề tài hướng đến các mục tiêu sau:

Xây dựng hệ thống thông tin xe buýt đa chức năng:

- Cung cấp thông tin chi tiết và cập nhật về các tuyến xe buýt, bao gồm lộ trình, lịch trình, giá vé, và các trạm dừng.

- Hiển thị vị trí xe buýt theo thời gian thực trên bản đồ, giúp người dùng dễ dàng theo dõi và lên kế hoạch di chuyển.
- Hỗ trợ tìm đường đi giữa hai điểm bất kỳ, bao gồm việc gợi ý các tuyến xe buýt cần đi và các điểm chuyển tuyến.

Thực hiện hóa trải nghiệm người dùng:

- Thiết kế giao diện trực quan, thân thiện với người dùng, dễ sử dụng trên cả máy tính và thiết bị di động.
- Cung cấp các tính năng tìm kiếm và lọc thông tin nhanh chóng, chính xác, giúp người dùng dễ dàng tìm thấy thông tin cần thiết.
- Hiển thị thông tin trực quan trên bản đồ, giúp người dùng nắm bắt được lộ trình và vị trí các trạm dừng một cách dễ dàng.

Ứng dụng công nghệ hiện đại:

- Sử dụng ReactJS để xây dựng giao diện người dùng động và phản hồi nhanh, đảm bảo trải nghiệm mượt mà trên các thiết bị khác nhau.
- Áp dụng Express.js để phát triển backend mạnh mẽ và linh hoạt, có khả năng xử lý nhiều yêu cầu đồng thời và tích hợp với các dịch vụ bên thứ ba.
- Tận dụng SQL Server để lưu trữ và quản lý dữ liệu hiệu quả, đảm bảo tính nhất quán và an toàn của thông tin.
- Triển khai WebSocket để cập nhật vị trí xe buýt theo thời gian thực, mang lại thông tin chính xác và kịp thời cho người dùng.

Đảm bảo hiệu suất và khả năng mở rộng:

- Thiết kế kiến trúc linh hoạt, dễ dàng mở rộng với các tính năng mới trong tương lai, như tích hợp thanh toán điện tử hoặc hệ thống đánh giá chất lượng dịch vụ.
- Đảm bảo hệ thống hoạt động ổn định và đáng tin cậy, với khả năng chịu lỗi và phục hồi nhanh chóng.

Thúc đẩy sử dụng giao thông công cộng:

- Cung cấp thông tin chính xác và dễ tiếp cận về hệ thống xe buýt, giúp người dùng tự tin hơn khi sử dụng phương tiện công cộng.
- Tích hợp các tính năng như gợi ý lộ trình tiết kiệm thời gian hoặc thân thiện với môi trường, khuyến khích người dùng chọn xe buýt thay vì phương tiện cá nhân.
- Xây dựng nền tảng cho việc thu thập dữ liệu về hành vi sử dụng xe buýt, giúp các cơ quan quản lý có thể hóa mạng lưới giao thông công cộng.

1.2 Giới thiệu các công nghệ sử dụng trong hệ thống

1.2.1 Giới thiệu về ReactJS

ReactJS là một thư viện JavaScript phổ biến được sử dụng để xây dựng giao diện

người dùng (UI) động và phản ứng nhanh chóng với các thay đổi trong dữ liệu. Dưới đây là lý do tại sao ReactJS là sự lựa chọn chính cho frontend:

Component-based Architecture (Kiến trúc dựa trên component):

ReactJS cho phép chia nhỏ giao diện thành các component độc lập và tái sử dụng, giúp phát triển giao diện một cách linh hoạt và dễ bảo trì. Ví dụ, các phần như giỏ hàng, danh sách tranh, thông tin chi tiết có thể được xây dựng thành các component riêng biệt, giúp dễ dàng tái sử dụng và cập nhật.

Virtual DOM (DOM ảo):

React sử dụng cơ chế Virtual DOM, giúp cải thiện hiệu suất khi thay đổi dữ liệu. Khi có sự thay đổi, React chỉ cập nhật những phần tử DOM cần thiết thay vì toàn bộ giao diện, giúp giảm thời gian render và mang đến trải nghiệm người dùng mượt mà.

React Router:

React Router được sử dụng để điều hướng giữa các trang khác nhau trên website (ví dụ: trang chủ, trang chi tiết sản phẩm, giỏ hàng, thanh toán). Điều này giúp xây dựng ứng dụng một trang (Single Page Application - SPA) mà không cần tải lại toàn bộ trang.

State Management (Quản lý trạng thái):

Để quản lý dữ liệu giữa các component, React sử dụng state và props. Đối với các ứng dụng phức tạp, có thể sử dụng các thư viện như Redux hoặc React Context API để quản lý trạng thái toàn cục (global state), ví dụ như thông tin người dùng đăng nhập, giỏ hàng, hay lịch sử đơn hàng.

Responsive Design (Thiết kế đáp ứng):

ReactJS kết hợp tốt với các công cụ như CSS Grid, Flexbox, và Media Queries để tạo ra một giao diện web có thể tự động điều chỉnh phù hợp với các thiết bị khác nhau (máy tính để bàn, máy tính bảng, điện thoại di động).

React Hooks

Hooks là một cải tiến lớn trong React, cho phép sử dụng state và các tính năng khác trong functional components mà không cần dùng class. Một số hooks phổ biến:

- `useState`: Quản lý state.
- `useEffect`: Thực hiện các tác vụ phụ như gọi API, cập nhật DOM.
- `useContext`: Chia sẻ dữ liệu giữa các components.

Unidirectional Data Flow:

- React sử dụng mô hình luồng dữ liệu một chiều (data flows down).
- Dữ liệu từ component cha được truyền xuống component con thông qua props.
- Điều này giúp quản lý dữ liệu trong ứng dụng dễ dàng và hạn chế lỗi.

Lợi ích của ReactJS

Hiệu suất cao

- Sử dụng Virtual DOM giúp React xử lý các thay đổi nhanh chóng và hiệu quả.

- Việc cập nhật DOM thật được giảm thiểu đáng kể.

Tái sử dụng mã

- Các thành phần được viết dưới dạng components giúp tái sử dụng dễ dàng.
- Điều này tiết kiệm thời gian, giảm thiểu lỗi và tăng năng suất lập trình.

Hệ sinh thái phong phú

- React không hoạt động độc lập mà có thể kết hợp với nhiều thư viện như Redux, React Router, Axios để xây dựng các ứng dụng phức tạp.

Dễ tích hợp

- React có thể tích hợp vào các ứng dụng hiện có mà không cần viết lại toàn bộ mã.
- Điều này lý tưởng cho việc nâng cấp hoặc mở rộng các ứng dụng cũ.

Ứng dụng thực tế của ReactJS

React được sử dụng trong nhiều lĩnh vực khác nhau, từ các ứng dụng nhỏ đến hệ thống lớn:

- Single Page Applications (SPA): Các ứng dụng chạy mượt mà, không cần tải lại trang.
- Mạng xã hội: Facebook, Instagram đều được xây dựng một phần bằng React.
- Dashboard: Các ứng dụng quản trị phức tạp, hiển thị dữ liệu thời gian thực.
- Ứng dụng di động: Với React Native, React được sử dụng để phát triển ứng dụng di động native.

1.2.2 Giới thiệu về ExpressJS

Express.js là một framework web mã nguồn mở, nhẹ và linh hoạt được xây dựng trên nền tảng Node.js, giúp đơn giản hóa quá trình phát triển các ứng dụng web và API. Được phát triển bởi TJ Holowaychuk và cộng đồng, Express.js đã trở thành một trong những framework phổ biến nhất trong hệ sinh thái Node.js nhờ vào thiết kế tối giản và hiệu suất cao.

Đặc điểm nổi bật của Express.js

- **Tối giản và nhẹ nhàng:** Express.js được thiết kế với triết lý tối giản, cung cấp các công cụ cần thiết để xây dựng ứng dụng web mà không áp đặt quá nhiều cấu trúc. Framework này chỉ cung cấp lớp mỏng các tính năng web cơ bản, cho phép nhà phát triển tự do thiết kế kiến trúc ứng dụng theo nhu cầu. Điều này khác biệt so với các framework như Rails hay Django, vốn có cách tiếp cận mang tính định hướng cao hơn.
- **Middleware:** Một trong những tính năng mạnh mẽ nhất của Express.js là hệ thống middleware. Middleware là các hàm có quyền truy cập vào đối tượng request, đối tượng response và hàm middleware tiếp theo trong chuỗi request-response. Middleware có thể được sử dụng để thực hiện nhiều tác vụ như xác thực người dùng, xử lý lỗi, ghi log, phân tích cú pháp dữ liệu request,

và nhiều nhiệm vụ khác.

- **Routing:** Express.js cung cấp hệ thống định tuyến mạnh mẽ cho phép ứng dụng phản hồi các yêu cầu HTTP khác nhau tại các URL khác nhau. Bạn có thể dễ dàng định nghĩa các route để xử lý các phương thức HTTP như GET, POST, PUT, DELETE, và các phương thức khác. Express.js cũng hỗ trợ các route có tham số, giúp xử lý các URL động một cách dễ dàng.
- **Hiệu suất cao:** Express.js được xây dựng trên Node.js, vốn nổi tiếng với mô hình I/O không đồng bộ (non-blocking) và khả năng xử lý nhiều kết nối đồng thời. Điều này giúp ứng dụng Express.js có hiệu suất cao, đặc biệt là trong các tình huống có nhiều yêu cầu đồng thời.

Ứng dụng của Express.js

- **RESTful API:** Express.js là một công cụ lý tưởng để xây dựng các API RESTful nhờ vào khả năng định tuyến linh hoạt và xử lý HTTP request/response một cách đơn giản. Nhiều doanh nghiệp lớn như Netflix, IBM, Uber và eBay đã sử dụng Express.js để xây dựng các API của họ.
- **Ứng dụng web:** Express.js có thể được sử dụng để xây dựng các ứng dụng web đầy đủ tính năng, từ các trang web tĩnh đơn giản đến các ứng dụng web động phức tạp. Kết hợp với các template engine, Express.js cho phép bạn tạo ra các giao diện người dùng động dựa trên dữ liệu từ server.
- **Ứng dụng thời gian thực:** Khi kết hợp với các thư viện WebSocket như Socket.IO, Express.js có thể được sử dụng để xây dựng các ứng dụng thời gian thực như ứng dụng chat, bảng điều khiển thời gian thực, hoặc các ứng dụng cộng tác.

Ưu điểm của Express.js

- Dễ học và sử dụng: Express.js có cú pháp đơn giản và dễ hiểu, giúp các nhà phát triển mới có thể nhanh chóng bắt đầu với framework này.
- Linh hoạt và tùy biến cao: Express.js không áp đặt cấu trúc cứng nhắc, cho phép nhà phát triển tự do thiết kế kiến trúc ứng dụng theo nhu cầu.
- Hiệu suất cao: Được xây dựng trên Node.js, Express.js thừa hưởng mô hình I/O không đồng bộ, giúp ứng dụng có khả năng xử lý nhiều kết nối đồng thời với hiệu suất cao.
- Hỗ trợ cộng đồng mạnh mẽ: Express.js có một cộng đồng lớn và tích cực, cung cấp nhiều tài nguyên, thư viện và middleware để giải quyết các vấn đề phổ biến.
- Tích hợp dễ dàng với cơ sở dữ liệu: Express.js có thể dễ dàng tích hợp với nhiều cơ sở dữ liệu như MongoDB, MySQL, PostgreSQL, và nhiều loại khác.

Nhược điểm của Express.js

- Thiếu cấu trúc chuẩn: Sự linh hoạt của Express.js đôi khi có thể dẫn đến mã nguồn không có cấu trúc rõ ràng, đặc biệt là trong các dự án lớn nếu không có

quy ước rõ ràng.

- Vấn đề với callbacks: Express.js sử dụng nhiều callbacks, có thể dẫn đến "callback hell" nếu không được quản lý cẩn thận. Tuy nhiên, vấn đề này có thể được giải quyết bằng cách sử dụng Promises hoặc async/await.
- Thông báo lỗi khó hiểu: Đôi khi các thông báo lỗi trong Express.js có thể khó hiểu, gây khó khăn cho việc gỡ lỗi.

Kết luận

Express.js là một framework web mạnh mẽ và linh hoạt cho Node.js, cung cấp các công cụ cần thiết để xây dựng các ứng dụng web và API một cách nhanh chóng và hiệu quả. Với thiết kế tối giản, hiệu suất cao và cộng đồng hỗ trợ mạnh mẽ, Express.js là lựa chọn lý tưởng cho các dự án từ nhỏ đến lớn. Trong dự án xây dựng trang web xe buýt của chúng ta, Express.js sẽ đóng vai trò quan trọng trong việc xây dựng backend mạnh mẽ, xử lý các yêu cầu từ frontend và tương tác với cơ sở dữ liệu SQL Server.

1.2.3 Giới thiệu về SQL Server

SQL Server là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) phát triển bởi Microsoft. Đây là một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất, đặc biệt trong môi trường doanh nghiệp và ứng dụng web quy mô lớn.

Đặc điểm chính của SQL Server:

- Hiệu năng cao: SQL Server được hóa để xử lý khối lượng dữ liệu lớn và truy vấn phức tạp một cách hiệu quả.
- Bảo mật mạnh mẽ: Cung cấp nhiều tính năng bảo mật như mã hóa dữ liệu, quản lý quyền truy cập, và kiểm toán.
- Tích hợp tốt với công nghệ Microsoft: Hoạt động mượt mà với các công nghệ khác của Microsoft như .NET Framework và Azure.
- Công cụ quản lý trực quan: SQL Server Management Studio (SSMS) cung cấp giao diện đồ họa để quản lý và truy vấn cơ sở dữ liệu.
- Hỗ trợ ACID: Đảm bảo tính toàn vẹn dữ liệu thông qua các thuộc tính Atomicity, Consistency, Isolation, và Durability.

Ứng dụng trong dự án xe buýt:

- Lưu trữ dữ liệu: SQL Server sẽ lưu trữ thông tin về tuyến xe, trạm dừng, lịch trình, và vị trí xe buýt.
- Truy vấn hiệu quả: Cho phép truy xuất nhanh chóng thông tin cần thiết, như tìm tuyến xe gần nhất hoặc thời gian đến của xe buýt.
- Bảo mật thông tin: Đảm bảo an toàn cho dữ liệu nhạy cảm như thông tin người dùng.
- Tích hợp với backend: Kết hợp tốt với Express.js thông qua các thư viện như Sequelize ORM.

1.2.4 Giới thiệu WebSocket

WebSocket là một giao thức truyền thông hai chiều, full-duplex trên một kết nối TCP duy nhất. Nó cho phép giao tiếp thời gian thực giữa client và server mà không cần polling liên tục.

Đặc điểm chính của WebSocket:

- Kết nối liên tục: Duy trì một kết nối mở giữa client và server.
- Giao tiếp hai chiều: Cho phép dữ liệu được gửi từ cả hai phía mà không cần yêu cầu mới.
- Độ trễ thấp: Lý tưởng cho các ứng dụng thời gian thực.
- Hiệu quả: Giảm tải cho server so với HTTP polling truyền thống.

Ứng dụng trong dự án xe buýt:

- Cập nhật vị trí xe buýt: Gửi thông tin vị trí xe buýt theo thời gian thực đến client.
- Thông báo: Gửi thông báo ngay lập tức về sự cố hoặc thay đổi lịch trình.
- Tương tác người dùng: Cho phép người dùng nhận thông tin cập nhật mà không cần làm mới trang.

1.2.5 Giới thiệu API bản đồ

API bản đồ là các giao diện lập trình ứng dụng cho phép tích hợp các tính năng bản đồ vào ứng dụng web hoặc di động. Các API phổ biến bao gồm Google Maps API, Mapbox, và OpenStreetMap.

Đặc điểm chính của API bản đồ:

- Hiển thị bản đồ: Cho phép nhúng bản đồ tương tác vào ứng dụng.
- Đánh dấu vị trí: Thêm các điểm đánh dấu cho trạm xe buýt hoặc vị trí xe.
- Vẽ đường: Hiển thị lộ trình xe buýt trên bản đồ.
- Geocoding: Chuyển đổi địa chỉ thành tọa độ và ngược lại.

Ứng dụng trong dự án xe buýt:

- Hiển thị bản đồ tương tác: Cho phép người dùng xem vị trí xe buýt và trạm dừng.
- Vẽ lộ trình: Hiển thị các tuyến xe buýt trên bản đồ.
- Hiển thị thông tin: Hiển thị thông tin chi tiết về trạm dừng và xe buýt khi người dùng tương tác với bản đồ.

1.2.6 Các công nghệ khác

Redux:

- Quản lý state cho ứng dụng React phức tạp.
- Giúp quản lý dữ liệu tập trung, dễ dàng debug và mở rộng.

Axios:

- Thư viện HTTP client cho browser và Node.js.
-
-

- Sử dụng để gửi các yêu cầu API từ React đến Express server.

JWT (JSON Web Tokens):

- Sử dụng cho xác thực và bảo mật API.
- Cho phép tạo token để xác thực người dùng giữa client và server.

Socket.io:

- Thư viện để triển khai WebSocket dễ dàng.
- Hỗ trợ fallback cho các trình duyệt không hỗ trợ WebSocket.

Sequelize:

- ORM (Object-Relational Mapping) cho Node.js.
- Giúp tương tác với SQL Server dễ dàng hơn thông qua JavaScript.
- Các công nghệ này sẽ hỗ trợ đắc lực trong việc xây dựng một ứng dụng web xe buýt hoàn chỉnh, đáp ứng các yêu cầu về hiệu suất, bảo mật và trải nghiệm người dùng.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Lấy yêu cầu hệ thống

2.1.1 Mô tả hệ thống bằng ngôn ngữ tự nhiên

Đồ án "Xây dựng trang web xe buýt" nhằm tạo ra một nền tảng trực tuyến cung cấp thông tin đầy đủ và chính xác về hệ thống xe buýt, giúp người dùng dễ dàng tìm kiếm thông tin về các tuyến xe, theo dõi vị trí xe buýt theo thời gian thực, và tìm đường đi giữa hai điểm bất kỳ. Hệ thống được thiết kế với hai vai trò chính: Người dùng thông thường và Quản trị viên. Mỗi vai trò có các chức năng riêng biệt, giúp hóa trải nghiệm người dùng và quản lý hệ thống một cách hiệu quả.

Chức năng

Người dùng thông thường là đối tượng chính sử dụng hệ thống, có thể tham gia vào các hoạt động tìm kiếm và theo dõi thông tin xe buýt. Các chức năng chính của người dùng bao gồm:

1. Tìm kiếm tuyến xe buýt:

- Người dùng có thể tìm kiếm tuyến xe buýt theo số hiệu tuyến, điểm đầu/cuối, hoặc các trạm dừng trên tuyến.
- Khi chọn một tuyến xe, người dùng sẽ được xem thông tin chi tiết về tuyến, bao gồm lộ trình, lịch trình, giá vé, và danh sách các trạm dừng.

2. Tìm đường đi:

- Người dùng có thể nhập điểm đi và điểm đến, hệ thống sẽ gợi ý các phương án di chuyển bằng xe buýt..
- Lộ trình được hiển thị trực quan trên bản đồ.

3. Theo dõi vị trí xe buýt theo thời gian thực:

- Người dùng có thể xem vị trí hiện tại của các xe buýt trên một tuyến cụ thể.
- Thông tin được cập nhật theo thời gian thực thông qua WebSocket, giúp người dùng ước tính thời gian chờ đợi.

4. Đăng ký và quản lý tài khoản:

- Người dùng có thể đăng ký tài khoản để lưu trữ các tuyến xe yêu thích, lịch sử tìm kiếm, và nhận thông báo về các thay đổi lịch trình.
- Quản lý thông tin cá nhân và cài đặt thông báo.

2.1.2 Lập bảng từ khóa

a) Xác định các actor của toàn hệ thống

Bảng 2.1 Danh sách actor trong hệ thống

STT	Tên actor	Mô tả
-----	-----------	-------

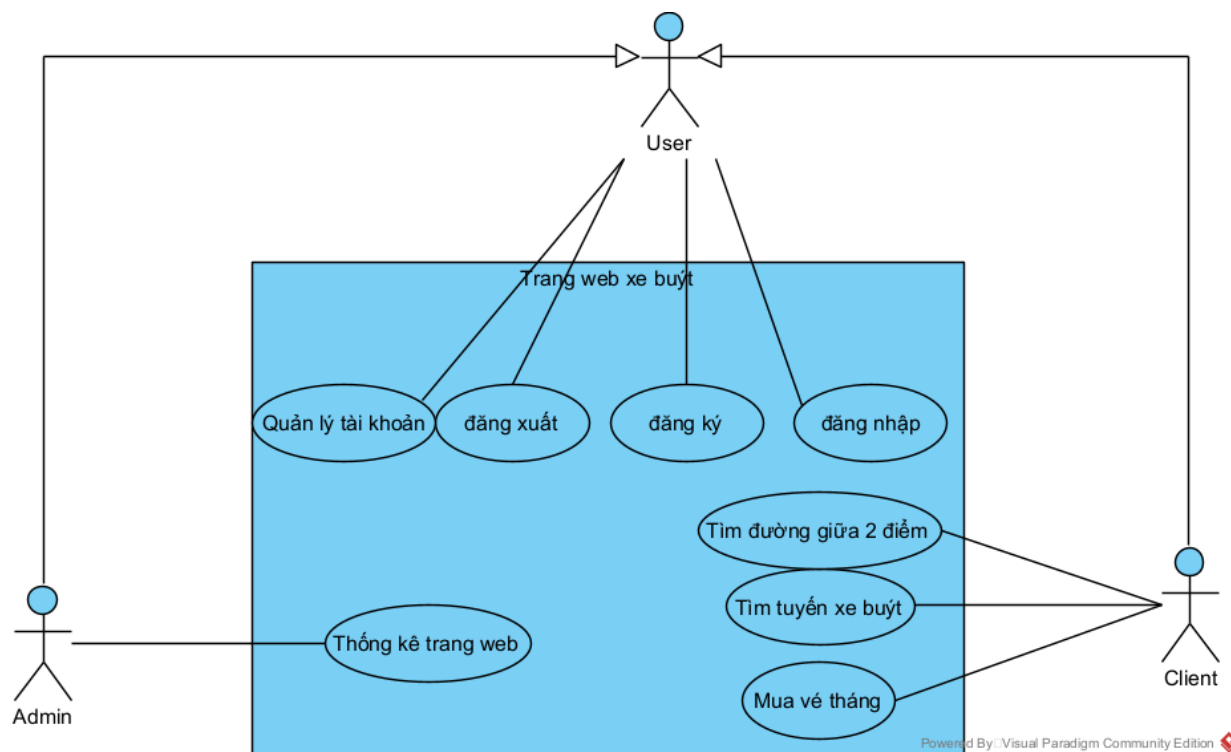
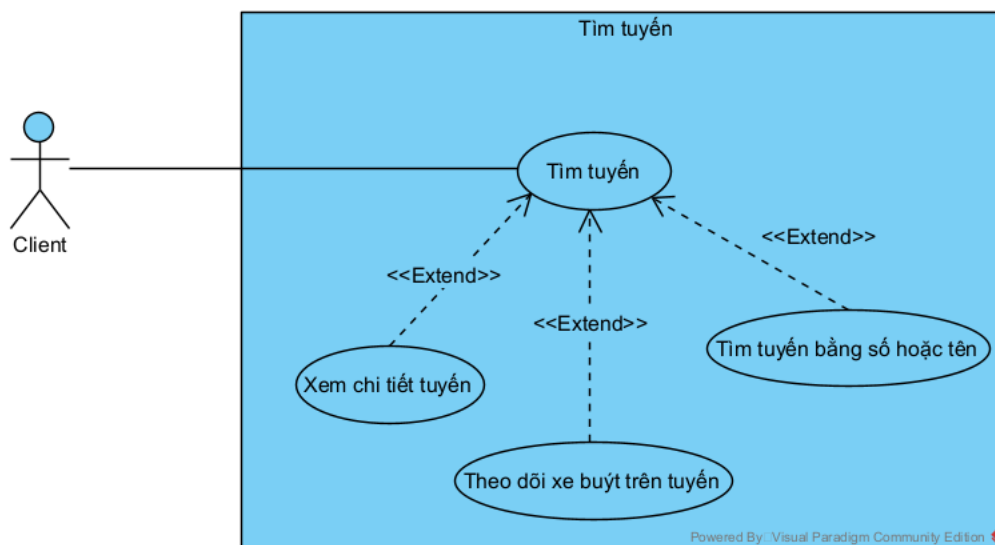
1	Người dùng (User)	Là tất cả người dùng sử dụng trang web
2	Khách (Client)	Là người dùng sử dụng trang web để thuận tiện cho việc tham gia giao thông bằng xe buýt
3	Quản trị viên(Admin)	Là người dùng có nhiệm vụ quản lý xe buýt và tài khoản hành khách

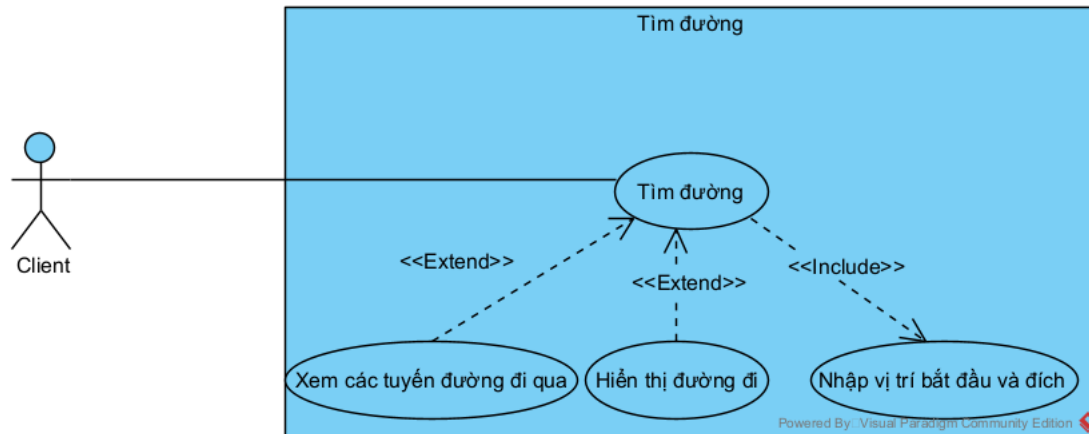
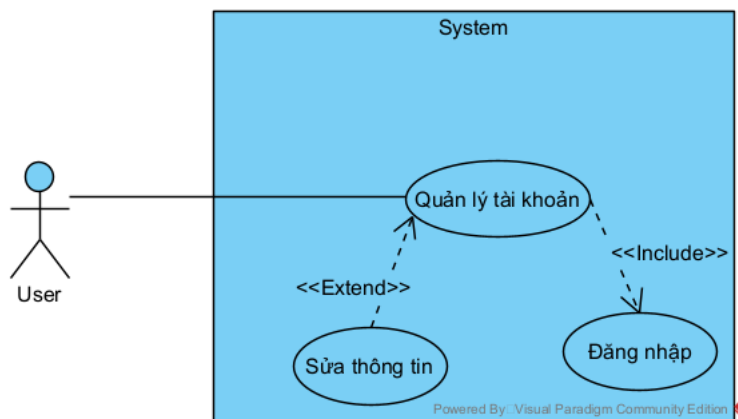
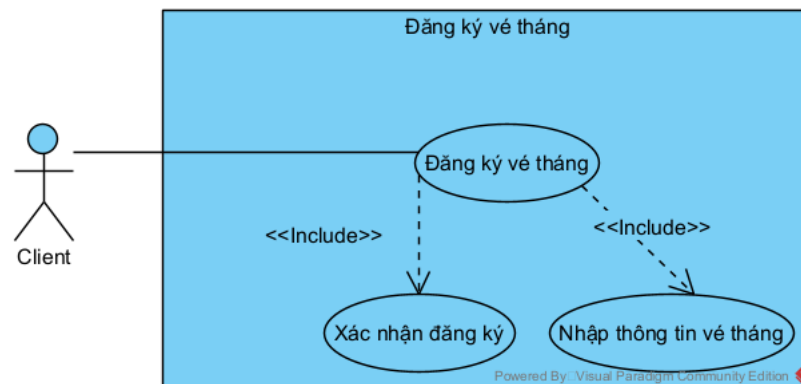
b) Xác định các usecase

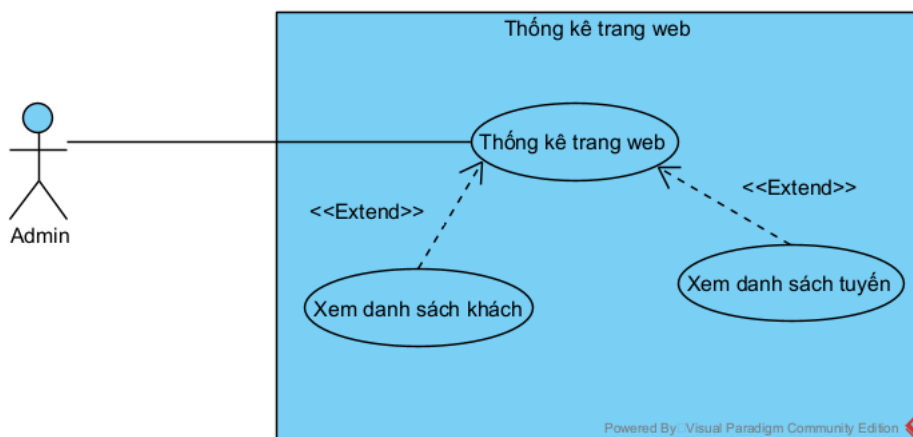
Bảng 2.2 Danh sách các usecase trong hệ thống

STT	Tên usecase	Mô tả
1	Tìm tuyến xe buýt	Người dùng có thể tìm, xem chi tiết thông tin tuyến và theo dõi xe buýt đi trên tuyến.
2	Tìm đường giữa 2 điểm	Người dùng có thể tìm các đường đi có thể từ vị trí bắt đầu đến kết thúc.
3	Quản lý tài khoản	Người dùng có thể xem, sửa thông tin cá nhân trên trang
4	Đăng ký vé tháng	Người dùng có thể đăng ký vé tháng xe buýt
5	Thông kê trang web	Người dùng có thể xem thống kê tuyến, bến xe buýt
6	Đăng nhập	Đăng nhập tài khoản người dùng có trong cơ sở dữ liệu
7	Đăng ký	Đăng ký tài khoản người dùng
8	Đăng xuất	Đăng xuất khỏi tài khoản

2.1.3 Biểu đồ usecase tổng quan

Biểu đồ use case toàn hệ thống**Hình 2.1: Biểu đồ usecase toàn hệ thống****2.1.4 Biểu đồ usecase chi tiết****Chức năng Tìm tuyến xe buýt**

Hình 2.2: Biểu đồ phân rã usecase Tìm tuyến xe buýt**Chức năng Tìm đường giữa 2 điểm****Hình 2.3: Biểu đồ phân rã usecase Tìm đường giữa 2 điểm****Chức năng Quản lý tài khoản****Hình 2.4: Biểu đồ phân rã usecase Quản lý tài khoản****Chức năng Đăng ký vé tháng**

Hình 2.5: Biểu đồ phân rã usecase Đăng ký vé tháng**Chức năng Thống kê trang web****Hình 2.6: Biểu đồ phân rã usecase Thống kê trang web****2.2 Phân tích hệ thống****2.2.1 Scenario****Kịch bản chức năng Tìm tuyến xe buýt****Bảng 2.3: Kịch bản chức năng Tìm tuyến xe buýt**

Tên use case	Tìm tuyến xe buýt
Actor	Client
Tiền điều kiện	
Đảm bảo thành công	Bản đồ hiển thị tuyến cần tìm
Chuỗi sự kiện chính:	<ol style="list-style-type: none"> 1. Người dùng truy cập vào trang web. 2. Hệ thống hiển thị giao diện trang web với thanh sidebar có thanh tìm kiếm cùng với danh sách các tuyến ở dưới và bản đồ.

	<div data-bbox="625 184 901 772"> <div> <div>Tìm tuyến</div> <div>Tìm đường</div> </div> <div> <div>Tìm kiếm tuyến</div> <div> <div>Tuyến 01</div> <div>Bến xe Gia Lâm - Bến xe Yên Nghĩa</div> <div>Chiều đi</div> </div> <div> <div>Tuyến 01</div> <div>Bến xe Yên Nghĩa - Bến xe Gia Lâm</div> <div>Chiều về</div> </div> <div> <div>Tuyến 02</div> <div>Bắc Cốt - Bến xe Yên Nghĩa</div> <div>Chiều đi</div> </div> <div> <div>Tuyến 02</div> <div>Bến xe Yên Nghĩa - Bắc Cốt</div> <div>Chiều về</div> </div> <div> <div>Tuyến 03A</div> <div>Bến xe Giáp Bát - Bến xe Gia Lâm</div> <div>Chiều đi</div> </div> </div> </div>
	<p>3. Người dùng nhập vào thanh tìm kiếm “01”.</p> <p>4. Hệ thống sẽ hiển thị thông tin 2 tuyến “01 chiều đi: Yên Nghĩa→Bx Gia Lâm” và “01 chiều về: Gia Lâm→Yên Nghĩa” ở sidebar.</p> <div data-bbox="625 982 1010 1381"> <div> <div>Tìm tuyến</div> <div>Tìm đường</div> </div> <div> <div>01</div> <div> <div>Tuyến 01</div> <div>Bến xe Gia Lâm - Bến xe Yên Nghĩa</div> <div>Chiều đi</div> </div> <div> <div>Tuyến 01</div> <div>Bến xe Yên Nghĩa - Bến xe Gia Lâm</div> <div>Chiều về</div> </div> </div> </div>
	<p>5. Người dùng ấn vào “01 chiều đi: Yên Nghĩa→Bx Gia Lâm”</p> <p>6. Hệ thống sẽ hiển thị ra thông tin tuyến xe buýt 01 chiều đi trên sidebar:</p>

Tìm tuyến

Tìm đường

Quay lại

Thông tin chi tiết

Số tuyến: 01

Tên tuyến: Bến xe Gia Lâm - Bến xe Yên Nghĩa

Thời gian hoạt động: 05:00 - 23:00

Giá vé: 10000 VND

Theo dõi xe buýt trên tuyến

Bến xe trên tuyến:

- Bến xe Gia Lâm
- 549 Nguyễn Văn Cừ (cột trước)
- 307 Nguyễn Văn Cừ
- 135 Nguyễn Văn Cừ
- E1.4 Điểm trung chuyển Long Biên

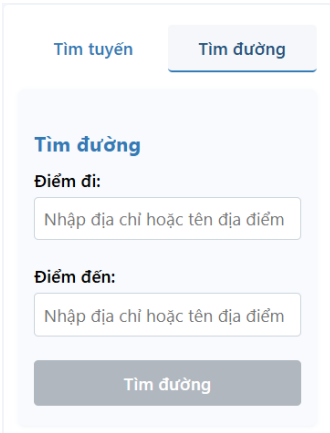
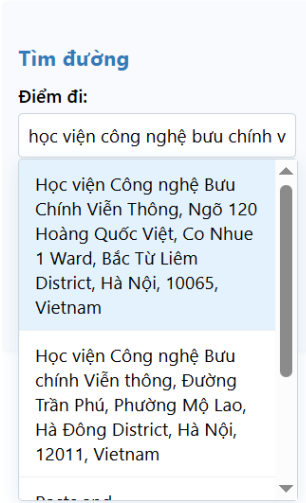
7. Người dùng ấn vào nút “Theo dõi xe buýt trên tuyến”

8. Hệ thống sẽ hiển thị tuyến đường đi và ký hiệu xe buýt di chuyển trên tuyến đường trên bản đồ:

Kịch bản chức năng Tìm đường giữa 2 điểm

Bảng 2.4: Kịch bản chức năng Tìm đường giữa 2 điểm

Tên use case	Tìm đường
Tác nhân chính	Client

Tiền điều kiện	
Đảm bảo thành công	Bản đồ hiển thị tuyến đường cần đi
Chuỗi sự kiện chính	<ol style="list-style-type: none"> Client nhấn chọn “Tìm đường” trên tab điều hướng. Hệ thống hiển thị 2 ô tìm kiếm điểm bắt đầu và điểm kết thúc  <ol style="list-style-type: none"> Client nhập vào ô tìm kiếm điểm bắt đầu “học viện công nghệ bưu chính viễn thông” Hệ thống sẽ hiển thị 1 danh sách lựa chọn tìm thấy:  <ol style="list-style-type: none"> Client ấn chọn vào lựa chọn “Posts and Telecommunications Institute of Technology, Km 10, Phố Nguyễn Văn Trỗi, Phường Mộ Lao, Hà Đông District, Hà Nội, 12011, Vietnam”

6. Hệ thống sẽ hiển thị icon đánh dấu vị trí bắt đầu trên bản đồ



7. Client nhập vào ô tìm kiếm điểm bắt đầu “ngõ 106 hoàng quốc việt”

8. Hệ thống sẽ hiển thị 1 danh sách lựa chọn tìm thấy:

Tìm đường

Điểm đi:

Posts and Telecommunications I

Điểm đến:

ngõ 106 hoàng quốc việt |

Ngõ 106 Hoàng Quốc Việt, Co
Nhue 1 Ward, Bắc Từ Liêm
District, Hà Nội, 10072,
Vietnam

Ngõ 106 Hoàng Quốc Việt, Co
Nhue 1 Ward, Bắc Từ Liêm
District, Hà Nội, 10065,
Vietnam

9. Client ấn chọn vào lựa chọn “Ngõ 106 Hoàng Quốc Việt, Co Nhue 1 Ward, Bắc Từ Liêm District, Hà Nội, 10072, Vietnam”

10. Hệ thống sẽ hiển thị icon đánh dấu vị trí bắt đầu trên bản đồ



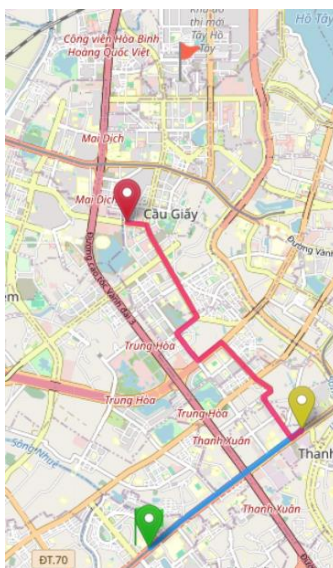
11. Client bấm vào nút “Tìm đường”.

12. Hệ thống sẽ hiển thị các danh sách đường đi có thể đi được ở sidebar:



13. Client ấn vào nút “Hiện thị trên đường đi” của hành trình đầu tiên

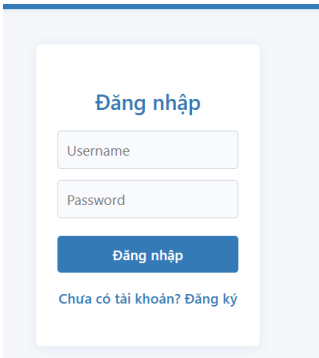

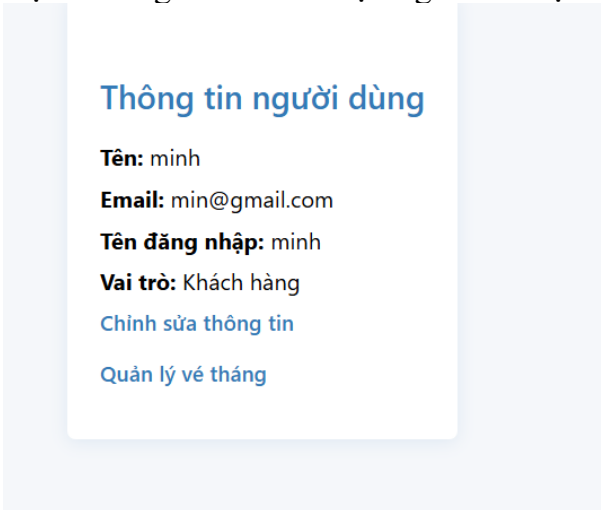
14. Hệ thống sẽ hiển thị đường đi trên bản đồ:



Kịch bản Quản lý tài khoản

Bảng 2.5: Kịch bản chức năng Quản lý tài khoản

Tên use case	Xem tài khoản
--------------	---------------

Tác nhân chính	User
Tiền điều kiện	User có tài khoản
Đảm bảo thành công	Hệ thống nhận sửa đổi thông tin
Chuỗi sự kiện chính	<ol style="list-style-type: none"> 1. User bấm vào nút đăng nhập trên header. 2. Hệ thống hiển thị giao diện đăng nhập với 2 ô nhập username, password với nút đăng nhập và đăng ký  <ol style="list-style-type: none"> 3. User nhập vào username = user123, password = 12345678. 4. Hệ thống hiển thị trang web nhưng header hiển thị nút tài khoản và đăng xuất thay vì đăng nhập và đăng ký  <ol style="list-style-type: none"> 5. User nhấn vào nút “Tài khoản” 6. Hệ thống hiển thị giao diện tài khoản 

	<p>7. User ấn vào nút “Chỉnh sửa thông tin”</p> <p>8. Màn hình hiển thị ra các vị trí nhập name, username, email, password và nút “Xác nhận”</p> <p>9. User nhập name=”Nguyễn Văn A”, username=”vana”, email=vana@gmail.com, password=”vanaomg123” và bấm nút xác nhận</p> <p>10. Hệ thống hiển thị ô thông báo đã cập nhật và nút OK</p> <p>11. User ấn vào nút OK</p> <p>12. Hệ thống trở về trang chủ</p>
Ngoại lệ	<p>4. Hệ thống thông báo thông “Sai tài khoản hoặc mật khẩu” và nút OK</p> <p>4.1. User ấn nút OK</p> <p>4.2. Hệ thống trở lại trang bước 2</p> <p>10. Hệ thống thông báo “Username đã tồn tại” và nút OK</p> <p>10.1. User ấn nút OK</p> <p>10.2. Hệ thống hiển thị bước 8</p>

Kịch bản Đăng ký vé tháng

Bảng 2.6: Kịch bản chức năng Đăng ký vé tháng

Tên use case	Đăng ký vé tháng
Tác nhân chính	Client
Tiền điều kiện	Client có tài khoản
Đảm bảo thành công	Hệ thống hiển thị mã QR
Chuỗi sự kiện chính	<p>1. Client bấm vào nút “Tài khoản” trên thanh header.</p> <p>2. Hệ thống hiển thị giao diện tài khoản.</p>

BusPer

Thông tin người dùng

Tên: Nguyễn Tuấn Minh
Email: Minhnt@gmail.com
Tên đăng nhập: user123

[Chỉnh sửa thông tin](#)
[Quản lý vé tháng](#)

3. Client nhấn vào nút “Quản lý vé tháng”
4. Hệ thống hiển thị giao diện mua vé tháng với ô nhập ngày bắt đầu

Vé tháng của tôi

Ngày bắt đầu

mm/dd/yyyy

Mua vé tháng

5. Client nhập 05/27/2025 và ấn nút Mua vé tháng

Vé tháng của tôi

Ngày bắt đầu

05/27/2025

Ngày bắt đầu: **2025-05-27**
Ngày kết thúc: **2025-06-26**
Giá vé: **200,000 VNĐ**

Mua vé tháng

6. Hệ thống hiển thị vé tháng bao gồm mã QR, thời gian hiệu lực, giá vé và trạng thái

Vé tháng của tôi

Vé tháng

Ngày bắt đầu: 2025-05-27T00:00:00.000Z
Ngày kết thúc: 2025-06-26T00:00:00.000Z
Giá vé: 200000 VNĐ

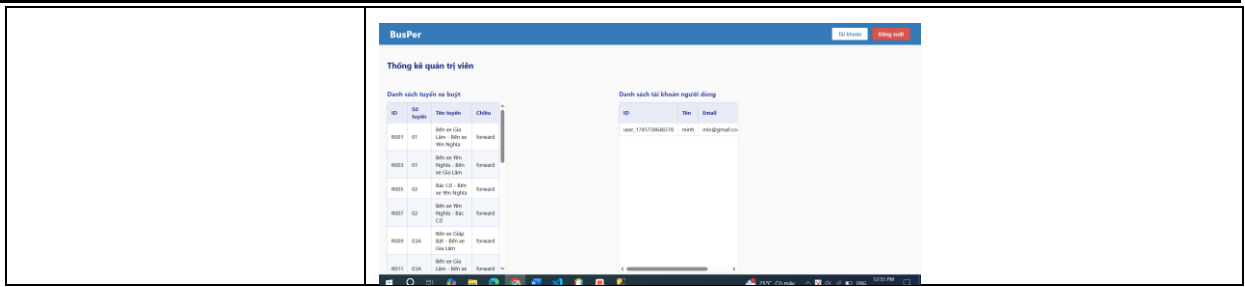
Trạng thái: Chưa đến ngày sử dụng



Kịch bản Thống kê trang web

Bảng 2.7: Kịch bản chức năng Thống kê trang web

Tên use case	Thống kê trang web
Tác nhân chính	Admin
Tiền điều kiện	Admin có tài khoản admin
Đảm bảo thành công	Hệ thống hiển thị danh sách client Hệ thống hiển thị danh sách tuyến xe buýt
Chuỗi sự kiện chính	<ol style="list-style-type: none"> Admin bấm vào nút đăng nhập trên header. Hệ thống hiển thị giao diện đăng nhập. Admin nhập vào username = “admin”, password = “admin123”. Hệ thống hiển thị trang web nhưng thanh header hiển thị nút thống kê, tài khoản và đăng xuất thay vì đăng nhập và đăng ký và <div data-bbox="630 1182 907 1220" data-label="Section-Header"> <h3>Thông tin người dùng</h3> </div> <div data-bbox="630 1234 876 1440" data-label="Text"> <p>Tên: Administrator Email: admin@example.com Tên đăng nhập: admin Vai trò: Quản trị viên Chỉnh sửa thông tin Trang thống kê quản trị viên</p> </div> Admin nhấn vào nút “Thống kê” Hệ thống hiển thị giao diện thống kê với 2 bảng hiển thị danh sách tuyến xe buýt và danh sách người dùng



2.3 Thiết kế hệ thống

2.3.1 Mô hình kiến trúc hệ thống

Hệ thống được thiết kế với 3 thành phần chính:

Ứng dụng (ReactJS):

- Là giao diện người dùng chính, phát triển bằng ReactJS. ReactJS tương tác trực tiếp với người dùng, gửi yêu cầu và nhận dữ liệu từ máy chủ Node.js/Express thông qua RESTful API hoặc WebSocket.
- ReactJS gửi các yêu cầu HTTP (GET, POST) đến Node.js/Express để xử lý logic nghiệp vụ và truy xuất dữ liệu từ SQL Server.
- Ứng dụng phục vụ các tài nguyên tĩnh (HTML, CSS, JavaScript) cho người dùng, đồng thời hiển thị bản đồ, vị trí xe buýt theo thời gian thực, thông tin tuyến, vé tháng, đánh giá, v.v.

Server (Node.js/Express):

- Là trung tâm xử lý dữ liệu và điều phối hệ thống.
- Node.js/Express tiếp nhận yêu cầu từ ứng dụng ReactJS qua RESTful API hoặc WebSocket, thực hiện xử lý logic nghiệp vụ và giao tiếp với cơ sở dữ liệu SQL Server.
- Node.js/Express cung cấp các API để ReactJS truy vấn dữ liệu (ví dụ: danh sách tuyến xe buýt, thông tin người dùng, bến xe, đánh giá, vé tháng, v.v.) và xử lý các yêu cầu khác từ người dùng.
- Node.js/Express bao gồm các chức năng như xác thực người dùng (JWT), xử lý form, quản lý đánh giá, mô phỏng vị trí xe buýt theo thời gian thực, và các tính năng khác.

Database (SQL Server):

- Lưu trữ dữ liệu của hệ thống, bao gồm thông tin người dùng, tuyến xe buýt, bến xe, xe buýt, đánh giá, vé tháng, v.v.

- Node.js/Express tương tác với SQL Server để thực hiện các tác vụ đọc/ghi dữ liệu thông qua thư viện mssql.
- SQL Server cung cấp dữ liệu cho Node.js/Express khi có yêu cầu API từ ReactJS.

Luồng hoạt động:

- Người dùng thao tác trên ReactJS và gửi yêu cầu (ví dụ: tìm tuyến xe buýt, đăng nhập, mua vé tháng, đánh giá tuyến, v.v.).
- ReactJS gửi yêu cầu HTTP hoặc WebSocket tới Node.js/Express qua RESTful API.
- Node.js/Express nhận yêu cầu từ ReactJS, xử lý logic nghiệp vụ và thực hiện các thao tác cần thiết với SQL Server (ví dụ: truy vấn thông tin tuyến, xác thực người dùng, lưu đánh giá).
- Node.js/Express trả kết quả về cho ReactJS dưới dạng JSON hoặc dữ liệu thời gian thực.
- ReactJS nhận kết quả từ Node.js/Express và hiển thị thông tin cho người dùng.

2.3.2 Biểu đồ lớp thực thể**User_account (Người dùng)**

Mô tả: Người dùng là trung tâm của hệ thống, đại diện cho hành khách sử dụng các chức năng như tra cứu tuyến xe buýt, tìm đường đi giữa 2 điểm, mua vé tháng, quản lý tài khoản.

Thuộc tính:

- id: Khóa chính định danh duy nhất mỗi người dùng.
- name: Họ tên của người dùng
- username: Tên đăng nhập
- password: Mật khẩu
- mail: Email
- role: Vai trò của tài khoản người dùng

Quan hệ:

- 1-1 với Monthly_ticket: Một người dùng có thể có một hoặc không có vé tháng

Monthly_ticket (Vé tháng)

Mô tả: Vé tháng là loại vé cho phép người dùng đi lại không giới hạn trên mọi tuyến

trong thời gian một tháng.

Thuộc tính:

- id: Khóa chính định danh vé tháng.
- start_date: Ngày bắt đầu
- end_date: Ngày hết hiệu lực
- qr_code: Mã qr
- price: Giá vé
- status: Trạng thái

Quan hệ:

- 1-1 với User_account.

Route (Tuyến xe buýt)

Mô tả: Tuyến xe buýt là đối tượng mô tả các tuyến đường mà xe buýt hoạt động.

Thuộc tính:

- id: Khóa chính định danh tuyến xe buýt.
- number: Số của tuyến xe buýt
- name: Tên của tuyến xe buýt
- direction: Chiều đi hoặc về của tuyến xe buýt
- distance: Độ dài tuyến
- ticket_price: Giá vé của tuyến
- unit: Đơn vị quản lý
- start_point: Điểm bắt đầu
- end_point: Điểm kết thúc
- operation_time: Giờ hoạt động
- frequency: Tần suất

Quan hệ:

- 1-N với Bus: Mỗi tuyến xe buýt có nhiều hơn một xe buýt.
- N-N với Node: thông qua bảng Route_node

Bus (Xe buýt)

Mô tả: Xe buýt là phương tiện vận chuyển hành khách trên các tuyến.

Thuộc tính:

- license_plate: Khóa chính định danh từng xe buýt.
 - latitude: Vĩ độ hiện tại.
 - longitude: Kinh độ hiện tại.
 - direction: Hướng di chuyển.
-
-

- speed: Tốc độ
- update_at: Thời gian cập nhật vị trí

Quan hệ:

- N-1 với Route: Mỗi xe buýt thuộc về một hoặc hai tuyến (đi/về).

Node(Điểm nút)

Mô tả: Node là các điểm trên bản đồ, có thể là bến xe buýt hoặc điểm chuyển.

Thuộc tính:

- id: Khóa chính định danh điểm nút
- latitude: Vĩ độ điểm nút
- longitude: Kinh độ điểm nút
- address: Địa chỉ nếu có
- is_bus_stop: Điểm chuyển này có phải bến xe buýt không

Quan hệ:

- N-N với Route qua bảng trung gian Route_node.

Route_node (Điểm chuyển trong tuyến)

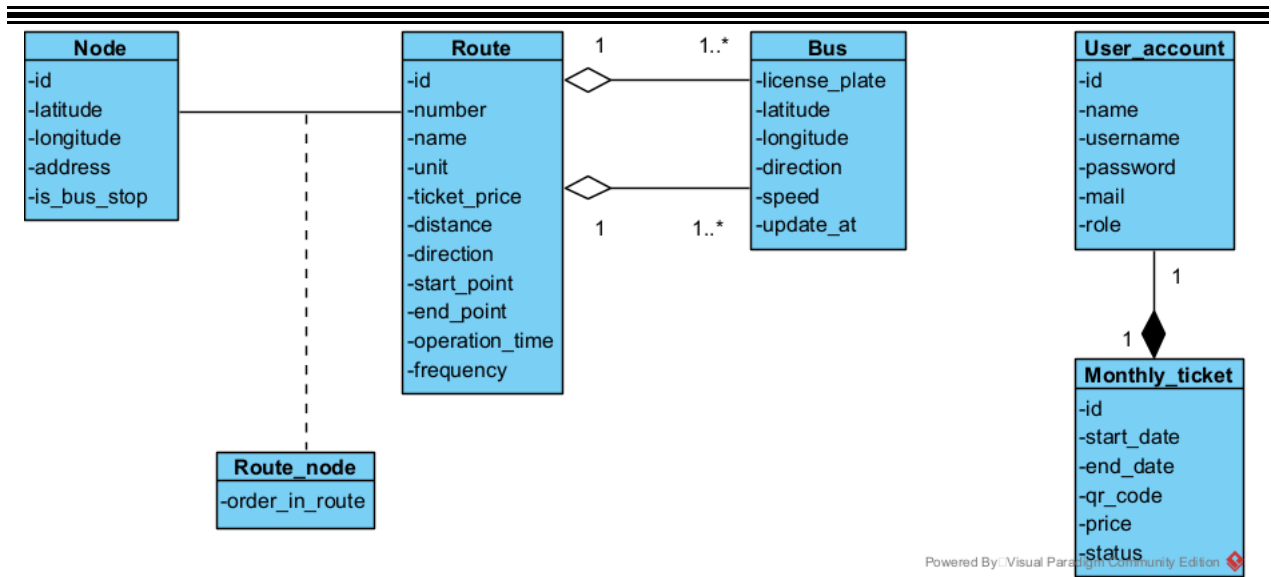
Mô tả: Bảng liên kết giữa tuyến và các điểm nút, xác định thứ tự các điểm trên tuyến.

Thuộc tính:

- Order_in_route: Thứ tự của các điểm chuyển trong tuyến

Quan hệ:

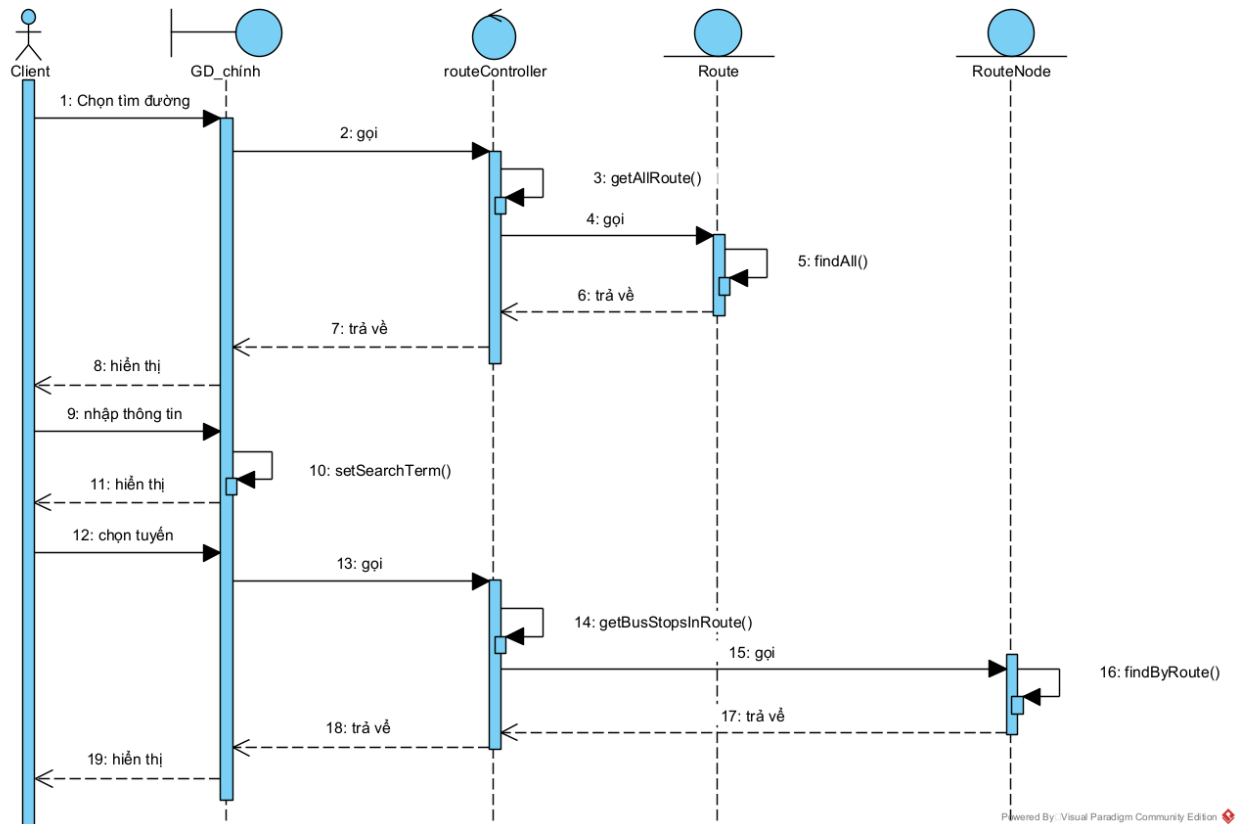
- N-1 với Route
 - N-1 với Node
-



Hình 2.7: Biểu đồ lớp thực thể

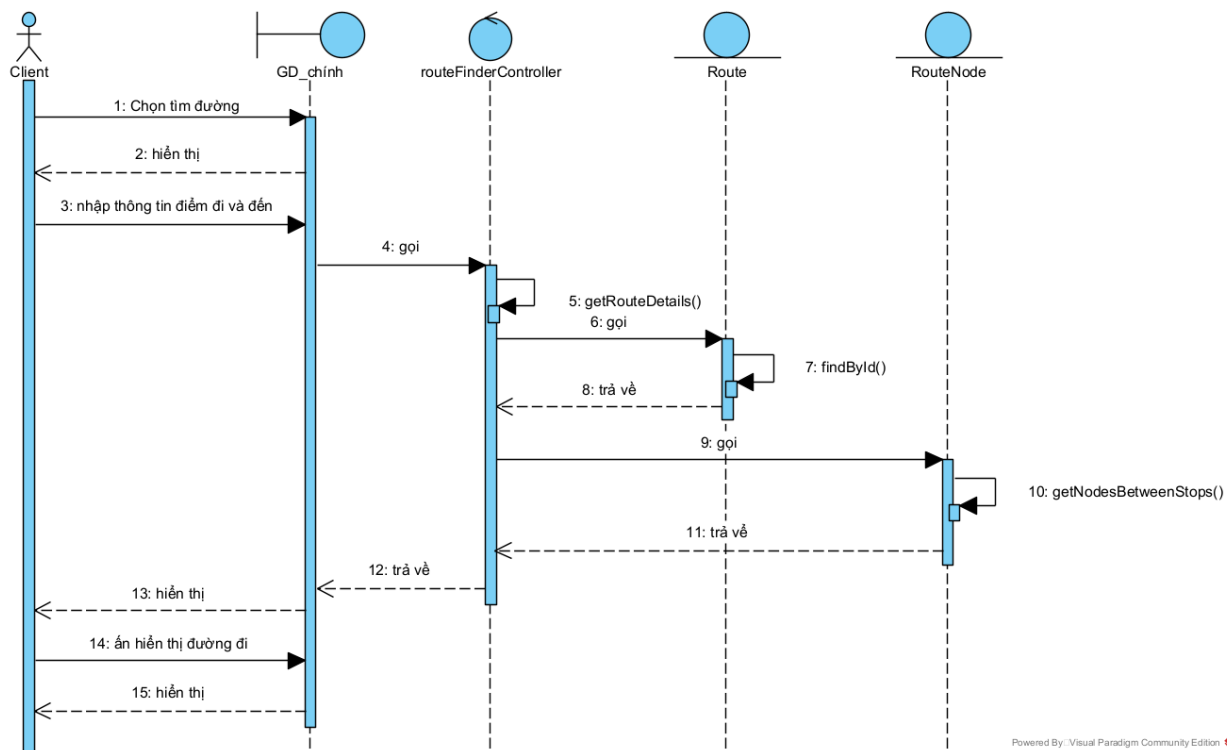
2.3.3 Biểu đồ tuần tự cho từng usecase

Chức năng Tìm tuyến xe buýt:

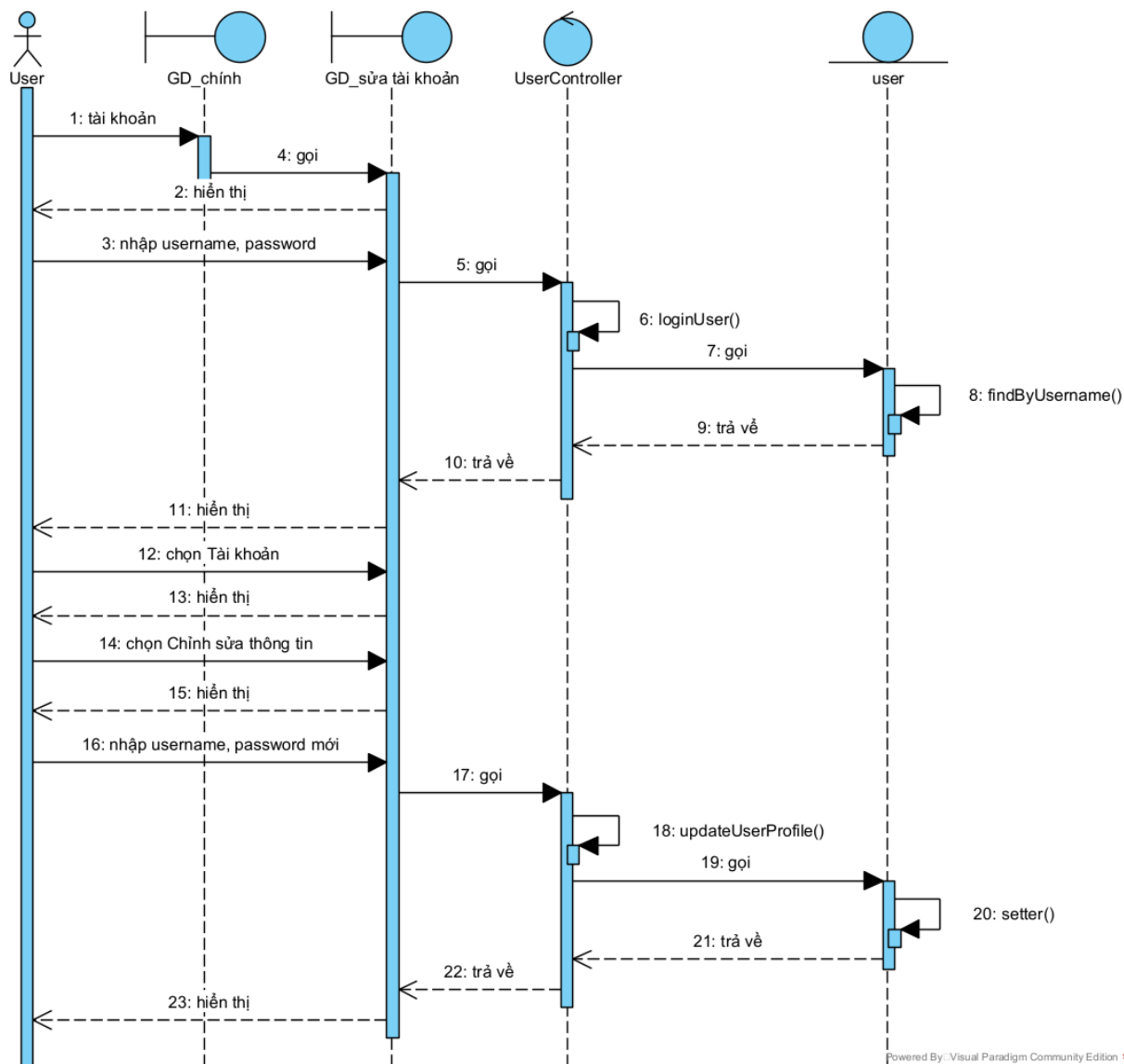


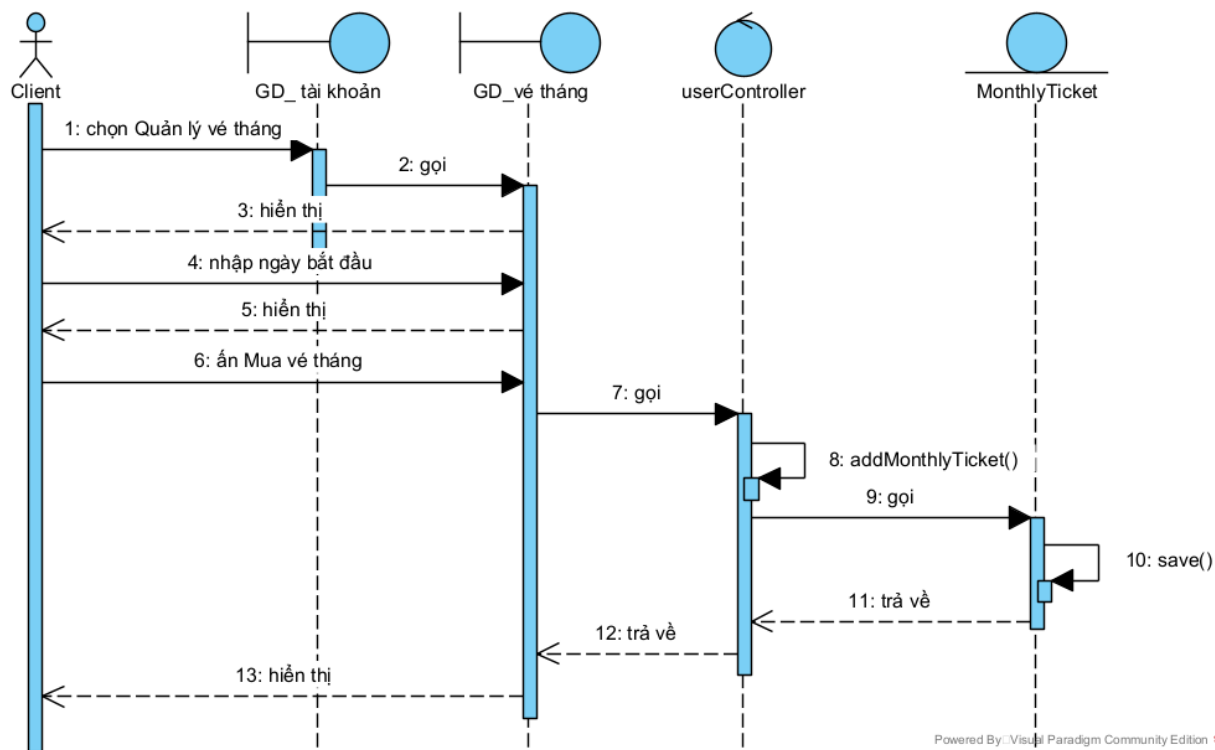
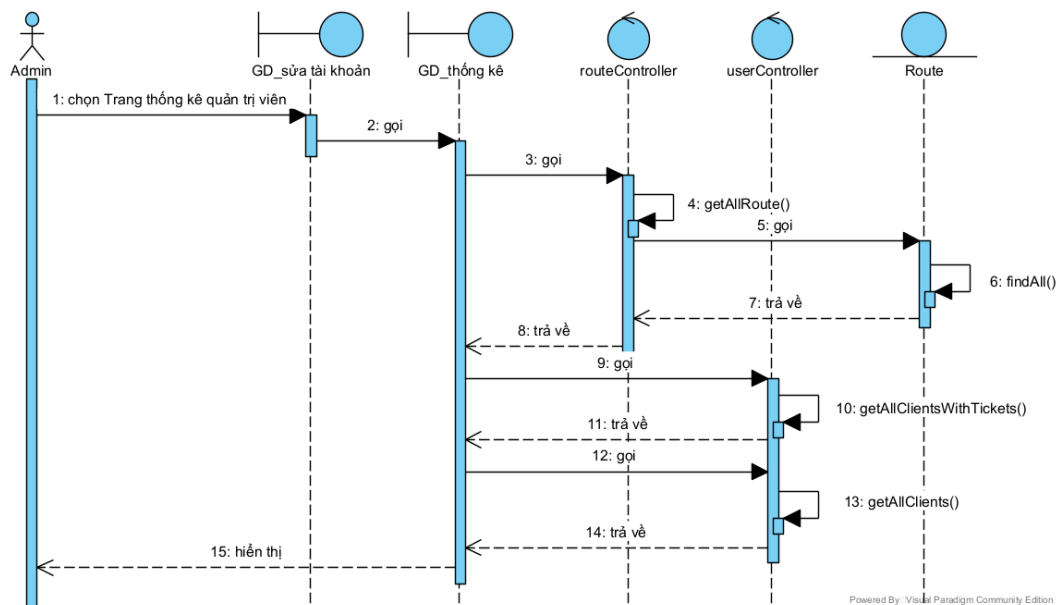
Hình 2.8: Sơ đồ tuần tự chức năng Tìm tuyến xe buýt

Chức năng tìm đường giữa 2 điểm



Hình 2.9: Sơ đồ tuần tự chức năng tìm đường giữa 2 điểm

Chức năng Xem tài khoản:**Hình 2.10: Sơ đồ tuần tự chức năng Quản lý tài khoản**

Chức năng Đăng ký vé tháng:**Hình 2.11: Sơ đồ tuần tự chức năng Đăng ký vé tháng****Chức năng Thống kê trang web****Hình 2.12: Sơ đồ tuần tự chức năng Thống kê trang web**

2.3.4 Thiết kế cơ sở dữ liệu

Mô tả chi tiết CSDL:

Bảng User_account: Lưu thông tin người sử dụng

Bảng 2.8: Mô tả bảng User_account

Tên cột	Kiểu Dữ liệu	Not null	Mặc định	Ràng buộc
id	Varchar(20)	Y	N/A	PK
name	Nvarchar(255)	Y	N/A	
username	Varchar(50)	Y	N/A	
password	Varchar(255)	Y	N/A	
mail	Varchar(255)	N	N/A	
role	Varchar(20)	Y	N/A	

Bảng Monthly_ticket: Lưu thông tin vé tháng

Bảng 2.9: Mô tả bảng Monthly_ticket

Tên cột	Kiểu dữ liệu	Not null	Mặc định	Ràng buộc
id	Varchar(50)	Y	N/A	PK
user_id	Varchar(20)	Y	N/A	FK
start_date	Date	Y	N/A	
end_date	Date	Y	N/A	
qr_code	Text	Y	N/A	
price	Int	Y	N/A	

status	Int	Y	N/A	
--------	-----	---	-----	--

Bảng Route : Lưu thông tin tuyến xe buýt**Bảng 2.10: Mô tả bảng Route**

Tên cột	Kiểu dữ liệu	Not null	Mặc định	Ràng buộc
id	Varchar(20)	Y	N/A	PK
Number	Varchar(20)	Y	N/A	
unit	Nvarchar(255)	Y	N/A	
name	Nvarchar(255)	Y	N/A	
ticket_price	Varchar(20)	Y	N/A	
distance	Varchar(20)	Y	N/A	
direction	Varchar(20)	Y	N/A	
start_point	Varchar(255)	Y	N/A	
end_point	Varchar(255)	Y	N/A	
operation_time	Varchar(50)	Y	N/A	
frequency	Varchar(50)	Y	N/A	

Bảng Bus: Lưu thông tin xe buýt**Bảng 2.11: Mô tả bảng Bus**

Tên cột	Kiểu dữ liệu	Not null	Mặc định	Ràng buộc
license_plate	Varchar(20)	Y	N/A	PK

route_id_forward	Varchar(20)	Y	N/A	FK
route_id_backward	Varchar(20)	Y	N/A	FK
latitude	Float	Y	N/A	
longitude	Float	Y	N/A	
direction	Float	Y	N/A	
speed	Float	Y	N/A	
update_at	datetime	Y	CURRE N_TIME STAMP	

Bảng Node: Lưu thông tin điểm nút**Bảng 2.12: Mô tả bảng Node**

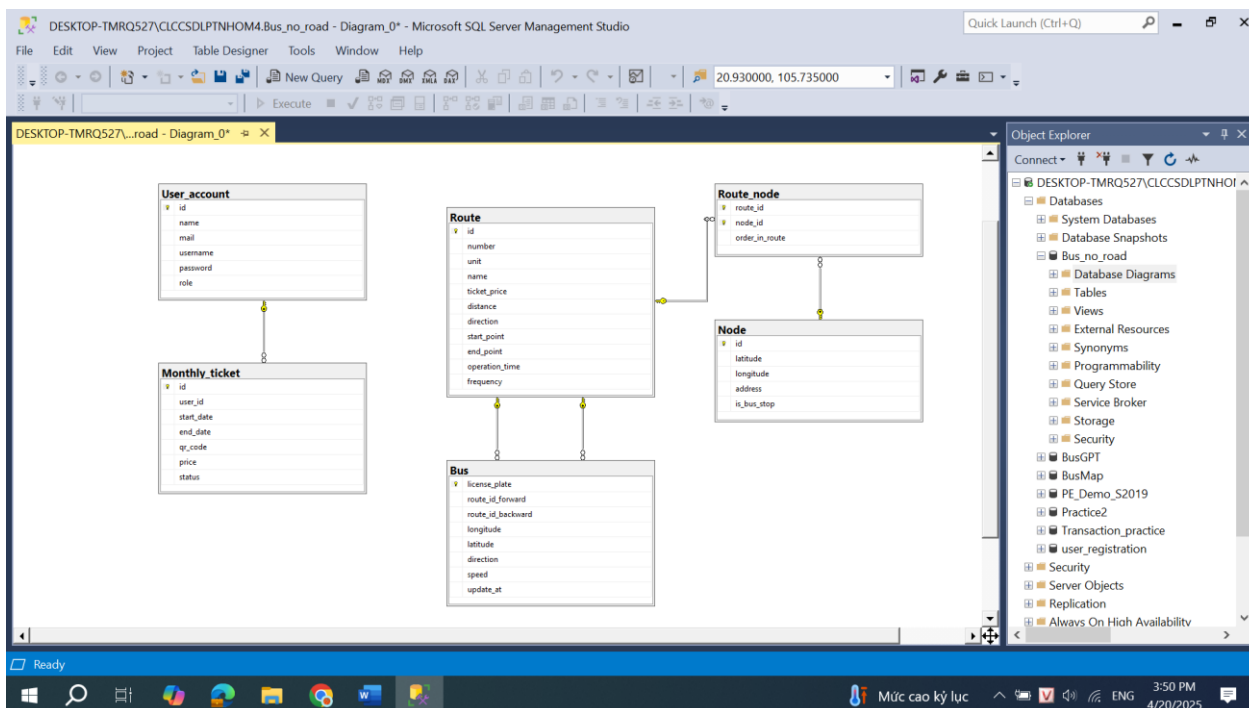
Tên cột	Kiểu dữ liệu	Not null	Mặc định	Ràng buộc
id	Varchar(20)	Y	N/A	PK
latitude	Float	Y	N/A	
longitude	Float	Y	N/A	
address	Nvarchar(255)	Y	N/A	
is_bus_stop	int	Y	N/A	

Bảng Route_node: Lưu thông tin điểm nút trong tuyến**Bảng 2.13: Mô tả bảng Route_node**

Tên cột	Kiểu dữ liệu	Not null	Mặc định	Ràng buộc

route_id	Varchar(20)	Y	N/A	PK, FK
node_id	Varchar(20)	Y	N/A	PK, FK
order_in_route	Int	Y	N/A	

2.3.5 Biểu đồ cơ sở dữ liệu



Hình 2.13: Biểu đồ cơ sở dữ liệu

CHƯƠNG 3: CÀI ĐẶT VÀ THỬ NGHIỆM

3.1 Cài đặt hệ thống và triển khai hệ thống

Yêu cầu công cụ:

- Tải và cài đặt NodeJs trên trang chủ của NodeJs:

<https://nodejs.org/en/download/package-manager>

Node.js là một môi trường chạy JavaScript mã nguồn mở, đa nền tảng, được xây dựng trên công cụ V8 JavaScript Engine của Google. Nó cho phép thực thi JavaScript phía máy chủ, không chỉ giới hạn trong trình duyệt, giúp xây dựng các ứng dụng nhanh, hiệu quả và dễ mở rộng.

Node.js sử dụng kiến trúc đơn luồng, bất đồng bộ và hướng sự kiện, giúp xử lý hàng nghìn kết nối đồng thời mà không bị chặn, tối ưu hóa hiệu suất. Với hệ thống quản lý gói mạnh mẽ npm (Node Package Manager), Node.js cung cấp hàng triệu thư viện hỗ trợ, từ đó giảm thiểu thời gian phát triển.

Node.js được ứng dụng phổ biến trong các lĩnh vực như xây dựng ứng dụng web thời gian thực (chat, game), API backend (RESTful, GraphQL), xử lý luồng dữ liệu streaming (phát nhạc, video), và hệ thống microservices. Sự đơn giản, hiệu suất và khả năng mở rộng làm cho Node.js trở thành lựa chọn hàng đầu của các nhà phát triển hiện đại.

- Tải và cài đặt visual studio

<https://code.visualstudio.com/download>

Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn miễn phí và mã nguồn mở, phát triển bởi Microsoft. Nó hỗ trợ nhiều ngôn ngữ lập trình, bao gồm JavaScript, Python, C++, và nhiều hơn nữa. VS Code có giao diện người dùng thân thiện, tính năng tự động hoàn thành mã, gỡ lỗi tích hợp, và khả năng mở rộng mạnh mẽ qua các tiện ích mở rộng (extensions). Đây là một công cụ phổ biến trong cộng đồng lập trình nhờ tính nhẹ, nhanh chóng và dễ sử dụng.

- Tải và cài đặt hệ quản trị cơ sở dữ liệu SQL Server

<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) được phát triển và duy trì bởi Microsoft. SQL Server sử dụng ngôn ngữ truy vấn SQL để quản lý dữ liệu, và là một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất trên thế giới, đặc biệt là trong các ứng dụng doanh nghiệp và web.

3.1.1 Cài đặt và triển khai phía BE

- Bước 1: Khởi tạo dự án Express.js và tải những thư viện cần thiết:

```
npm install express mssql dotenv cors bcrypt jsonwebtoken socket.io  
@turf/turf qrcode
```

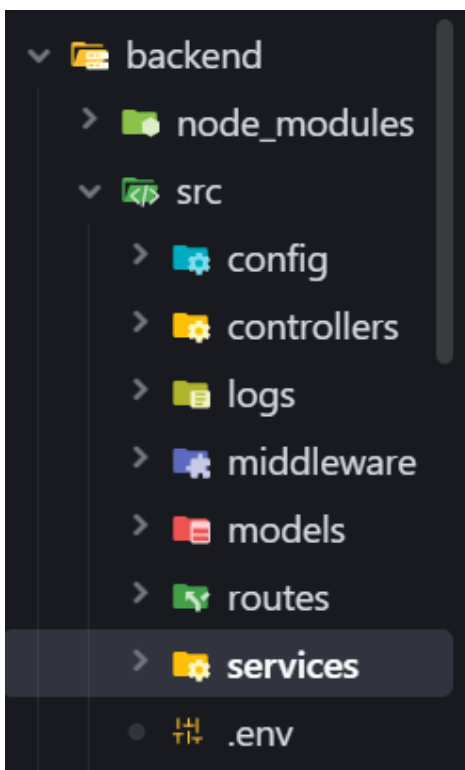
- Bước 2: Cấu hình biến môi trường .env

```

• PORT=3000
• DB_USER=sa
• DB_PASSWORD=123456
• DB_SERVER=DESKTOP-TMRQ527
• DB_NAME=Bus_no_road
• JWT_SECRET=pk8
• JWT_EXPIRES_IN=24h

```

- Bước 3: Xây cây thư mục và các file cần thiết.



3.1.2 Cài đặt Database

- Bước 1: Tạo SQL Server database
- Bước 2: Kết nối data base với backend qua file môi trường.env
- Bước 3: Tạo các bảng dữ liệu

```

• create table Route(
•   id varchar(20) primary key, -- Id cho mỗi tuyến (cả chuyến đi và về)
•   number varchar(20), -- số tuyến: 01, 02, 03A, 03B, ...
•   unit nvarchar(255), -- thuộc công ty nào
•   name nvarchar(255), -- ví dụ: bến xe Gia Lâm - bến xe Yên Nghĩa

```

```

•   ticket_price varchar(20), -- giá vé
•   distance varchar(20), -- độ dài tuyến
•   direction varchar(20), -- forward hoặc backward (đánh dấu chuyển đi hay
chuyển về của cùng số tuyến: 01, 02, ...)
•   start_point nvarchar(255), -- điểm bắt đầu của tuyến xe buýt
•   end_point nvarchar(255), -- điểm kết thúc của tuyến xe buýt
•   operation_time nvarchar(50), -- thời gian hoạt động của tuyến xe buýt
•   frequency nvarchar(50) -- tần suất chạy xe của tuyến
• );
• create table Bus(
•   license_plate varchar(20) primary key, --biển số xe
•   route_id_forward varchar(20), --id tuyến chuyển đi
•   route_id_backward varchar(20), -- id tuyến chuyển về (cần cùng số tuyến với
chuyển đi)
•   longitude float, --kinh độ
•   latitude float, --vĩ độ
•   direction float, --hướng xe đang đi
•   speed float, -- tốc độ
•   update_at datetime DEFAULT CURRENT_TIMESTAMP,
•   constraint fk_bus_route_forward foreign key (route_id_forward) references
Route(id),
•   constraint fk_bus_route_backward foreign key (route_id_backward) references
Route(id)
• );
• create table User_account(
•   id varchar(20) primary key, --id tài khoản
•   name nvarchar(255), --tên người dùng
•   mail varchar(255), -- email
•   username varchar(50), --tên tài khoản
•   password varchar(255), --mật khẩu
•   role varchar(20) default 'client' -- client hoặc admin
• );
• create table Monthly_ticket(
•   id varchar(50) primary key,
•   user_id varchar(20),
•   start_date date,
•   end_date date,
•   qr_code nvarchar(max),
•   price int,

```

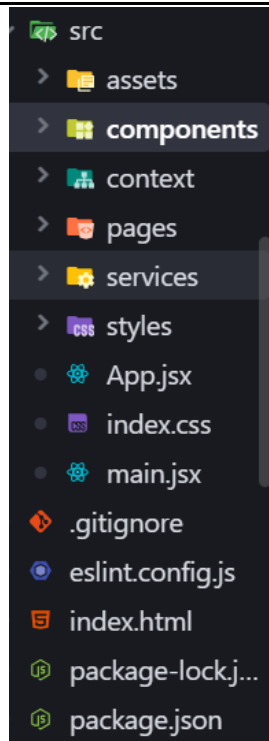
```

•     status varchar(20),
•     constraint fk_ticket_user foreign key (user_id) references User_account(id)
• );
• create table Node(
•     id varchar(20) primary key, -- id điểm nối
•     latitude float, -- vĩ độ
•     longitude float, -- kinh độ
•     unique (latitude, longitude),
•     address nvarchar(225) null, -- nếu điểm nối là bến xe thì có địa chỉ
•     is_bus_stop int check (is_bus_stop in (0, 1)) --điểm nối có phải là bến xe
không
• );
• create table Route_node(
•     route_id varchar(20), -- id tuyến (vì chuyển đi và về đi 2 đường khác nhau)
•     node_id varchar(20), -- id điểm nối
•     order_in_route int, -- thứ tự của bến xe trong tuyến để tạo đường đi
liên tục cho xe buýt
•     primary key(route_id, node_id),
•     constraint fk_routebs_bs foreign key (node_id) references Node(id),
•     constraint fk_routebs_route foreign key (route_id) references route(id)
• );

```

3.1.3 Cài đặt và triển khai phía FE

- Bước 1: Khởi tạo dự án React Vite: `npm create vite@latest`
- Bước 2: Cài đặt các thư viện cần thiết: `npm install axios leaflet react-leaflet socket.io-client lodash.debounce`
- Bước 3: Cấu trúc thư mục và các file chính



- Bước 4: Thiết lập router và các thành phần giao diện
- Bước 5: Tích hợp các dịch vụ và thư viện

3.1.4 Thuật toán tìm đường BFS

Bước 1: Xây dựng cấu trúc đồ thị

Mô hình hóa các tuyến xe buýt dưới dạng đồ thị

- Các bến xe buýt là các điểm nút
- Các tuyến xe buýt là các cạnh

```
// Xây dựng đồ thị có hướng kết nối giữa các tuyến xe
const routeConnectionsResult = await pool.request().query(`
    WITH RouteStops AS (
    SELECT
        rn.route_id,
        rn.node_id,
        rn.order_in_route,
        r.direction
    FROM Route_node rn
    JOIN Route r ON rn.route_id = r.id
    )

```

```

•      SELECT DISTINCT
•          rs1.route_id as route1,
•          rs2.route_id as route2,
•          rs1.direction as direction1,
•          rs2.direction as direction2,
•          rs1.order_in_route as order1,
•          rs2.order_in_route as order2
•      FROM RouteStops rs1
•      JOIN RouteStops rs2 ON rs1.node_id = rs2.node_id
•      WHERE rs1.route_id <> rs2.route_id
•      AND rs1.node_id = rs2.node_id
•  `);
•
•  // Tạo đồ thị có hướng kết nối giữa các tuyến
•  const routeGraph = {};
•  routeConnectionsResult.recordset.forEach(conn => {
•      if (!routeGraph[conn.route1]) routeGraph[conn.route1] = [];
•      routeGraph[conn.route1].push({
•          routeId: conn.route2,
•          direction: conn.direction2,
•          transferOrder: conn.order2
•      });
•  });
•  });

```

Bước 2: Lấy các dữ liệu cần thiết

Sử dụng các model để truy xuất dữ liệu từ cơ sở dữ liệu:

- Node: Lấy danh sách tất cả các bến xe buýt hoặc điểm nối
- RouteNode: Lấy danh sách các điểm nối và bến xe buýt giữa các bến trên từng tuyến.
- Route: Lấy thông tin về các tuyến xe buýt.

Bước 3: Cài đặt thuật toán BFS

- Đầu vào: Tọa độ điểm bắt đầu và kết thúc do người dùng chọn.
- Chuyển đổi tọa độ sang node gần nhất (bến xe buýt).

```

•  const findNearestBusStops = async (lat, lng, limit = 1) => {
•      const pool = await connectDB();
•      // Sử dụng haversine để tính khoảng cách
•      const result = await pool.request()
•          .input('lat', sql.Float, lat)

```

```

• .input('lng', sql.Float, lng)
• .input('limit', sql.Int, limit)
• .query(`
•     SELECT TOP (@limit) id, latitude, longitude, address,
•         (6371 * ACOS(
•             COS(RADIANS(@lat)) *
•             COS(RADIANS(latitude)) *
•             COS(RADIANS(longitude) - RADIANS(@lng)) +
•             SIN(RADIANS(@lat)) *
•             SIN(RADIANS(latitude))
•         )) AS distance
•     FROM Node
•     WHERE is_bus_stop = 1
•     ORDER BY distance ASC
• `);
• return result.recordset;
• };

```

- Sử dụng thuật toán BFS để tìm đường với số lần chuyển tuyến ngắn nhất (tối đa 3) từ node bắt đầu đến node kết thúc
 - Khởi tạo hàng đợi với node bắt đầu.
 - Đánh dấu các node đã duyệt để tránh lặp lại.
 - Với mỗi node, đưa các node kề (có kết nối trực tiếp qua một tuyến nào đó) vào hàng đợi nếu chưa duyệt.
 - Kết thúc khi tìm thấy node đích.

```

• // BFS để tìm đường đi
• const allPaths = [];
• const queue = [];
• const visited = new Set();
•
• const startStopIds = startStops.map(s => s.id);
• const endStopIds = endStops.map(s => s.id);
•
• // Khởi tạo với các tuyến đi qua điểm bắt đầu
• startRoutes.forEach(route => {
•     const nearestStartStop = findNearestStopInRoute(route, startStops[0]);
•
•     if (nearestStartStop.stop) {
•         queue.push({

```

```

•         routes: [route],
•         stops: [nearestStartStop.stop],
•         transferStops: [],
•         totalDistance: nearestStartStop.distance
•     });
•     visited.add(route);
• }
• });
•
• while (queue.length > 0 && allPaths.length < maxRoutes) {
•     const current = queue.shift();
•     const currentRoute = current.routes[current.routes.length - 1];
•
•     // Nếu tuyến hiện tại đi qua điểm kết thúc
•     if (endRoutes.has(currentRoute)) {
•         const nearestEndStop = findNearestStopInRoute(currentRoute,
endStops[0]);
•
•         if (nearestEndStop.stop) {
•             const totalDistance = current.totalDistance +
nearestEndStop.distance;
•
•             allPaths.push({
•                 routes: current.routes,
•                 stops: [...current.stops, ...current.transferStops,
nearestEndStop.stop],
•                 totalDistance,
•                 transfers: current.routes.length - 1
•             });
•
•             continue;
•         }
•     }
•
•     if (current.routes.length > maxTransfers + 1) continue;
•
•     // Thêm các tuyến kề vào hàng đợi
•     (routeGraph[currentRoute] || []).forEach(nextRoute => {
•         if (!visited.has(nextRoute) || current.routes.length <= 1) {

```

```

•         let transferStops = findTransferStops(currentRoute,
nextRoute);
•
•         transferStops = transferStops.filter(
•             stop => !startStopIds.includes(stop.id) &&
!endStopIds.includes(stop.id)
•         );
•
•         if (transferStops.length > 0) {
•             const transferStop = transferStops[0];
•
•             // Tính khoảng cách từ bến cuối cùng đến điểm chuyển tuyến
•             const lastStop = current.stops[current.stops.length - 1];
•             const distanceToTransfer = calculateDistance(
•                 lastStop.latitude, lastStop.longitude,
•                 transferStop.latitude, transferStop.longitude
•             );
•
•             // Tính khoảng cách của tuyến tiếp theo
•             const nextRouteDistance =
calculateRouteDistance(nextRoute);
•
•             queue.push({
•                 routes: [...current.routes, nextRoute],
•                 stops: [...current.stops],
•                 transferStops: [...current.transferStops,
transferStop],
•                 totalDistance: current.totalDistance +
distanceToTransfer + nextRouteDistance
•             });
•
•             visited.add(nextRoute);
•         }
•     }
• });

```

- Lưu lại đường đi (danh sách các node) tìm được bởi BFS.

Bước 4: Xây dựng kết quả trả về

- Lấy thông tin chi tiết cho từng node trong đường đi

- Trả về đường đi dưới dạng danh sách các bến và các đoạn tuyến cho frontend hiển thị lên bản đồ

```

• exports.findRoute = async (req, res) => {
•   const { startPoint, endPoint } = req.body;
•   try {
•     // Tìm 3 bến gần nhất cho cả điểm bắt đầu và điểm kết thúc
•     const nearestStartStops = await
findNearestBusStops(startPoint.lat, startPoint.lng, 3);
•     const nearestEndStops = await findNearestBusStops(endPoint.lat,
endPoint.lng, 3);
•
•     if (nearestStartStops.length === 0 || nearestEndStops.length ===
0) {
•       return res.status(404).json({
•         message: 'Không tìm thấy bến xe buýt gần điểm xuất phát
hoặc điểm đến'
•       });
•     }
•
•     // Truyền cả 3 bến vào thuật toán tìm đường
•     let routes = await findRoutesWithBusRoutesBFS(nearestStartStops,
nearestEndStops, 10, 2);
•
•     const seen = new Set();
•     routes = routes.filter(route => {
•       const stopSeq = route.stops.map(stop => stop.id).join('-');
•       if (seen.has(stopSeq)) return false;
•       seen.add(stopSeq);
•       return true;
•     });
•
•     // Lấy thông tin chi tiết về các tuyến và các node theo từng đoạn
•     for (const route of routes) {
•       const details = [];
•       for (const routeId of route.routes) {
•         details.push(await getRouteDetails(routeId));
•       }
•       route.routeDetails = details;
•     }

```

```

    •     route.segments = [];
    •     for (let i = 0; i < route.stops.length - 1; i++) {
    •         const fromStop = route.stops[i];
    •         const toStop = route.stops[i + 1];
    •         const routeId = route.routes[Math.min(i,
route.routes.length - 1)];
    •         const nodes = await getNodesBetweenStops(routeId,
fromStop.id, toStop.id);
    •         route.segments.push({
    •             routeId,
    •             from: fromStop.id,
    •             to: toStop.id,
    •             nodes
    •         });
    •     }
    • }
    •
    •
    •     res.json({
    •         routes,
    •         startStops: nearestStartStops,
    •         endStops: nearestEndStops
    •     });
    • } catch (err) {
    •     console.error('Lỗi tìm đường:', err);
    •     res.status(500).json({ error: err.message });
    • }
    •
    • };

```

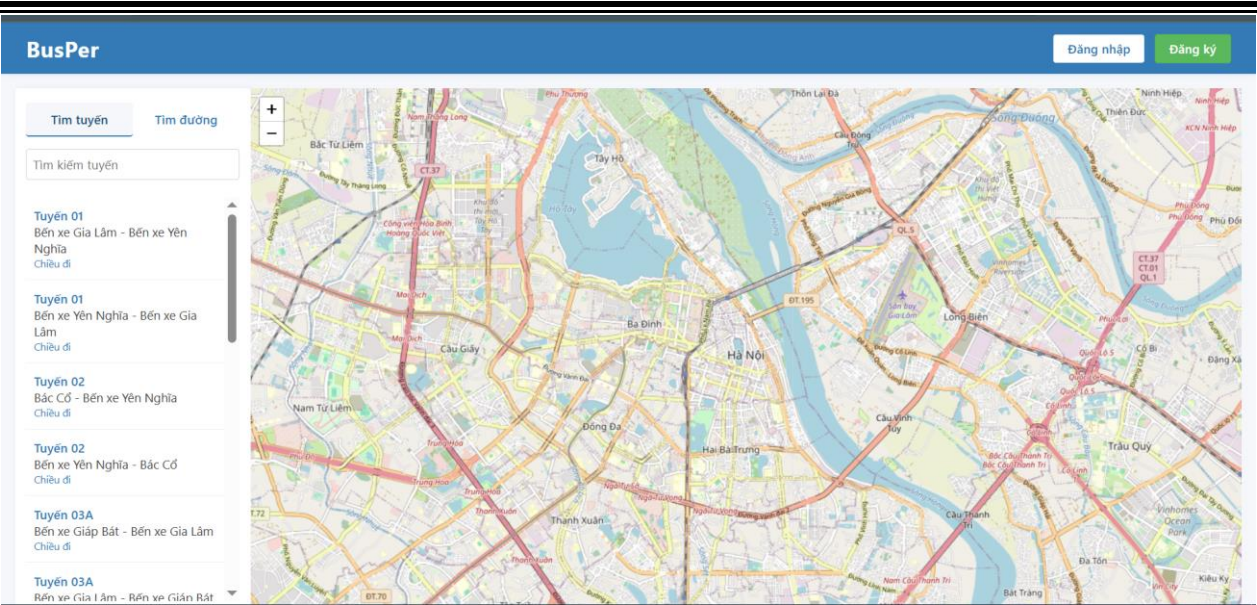
3.2 Kết luận

3.2.1 Kết quả đạt được

Qua quá trình thực hiện đồ án, hệ thống hỗ trợ tìm đường đi bằng xe buýt đã hoàn thiện các chức năng chính, bao gồm tìm kiếm tuyến xe, hiển thị lộ trình trên bản đồ, tra cứu thông tin bến xe, quản lý tài khoản người dùng, và mua vé tháng trực tuyến. Hệ thống đã tích hợp các công nghệ hiện đại như ReactJS cho frontend, Node.js/Express cho backend và SQL Server cho cơ sở dữ liệu, giúp tối ưu hóa hiệu suất và đảm bảo bảo mật dữ liệu.

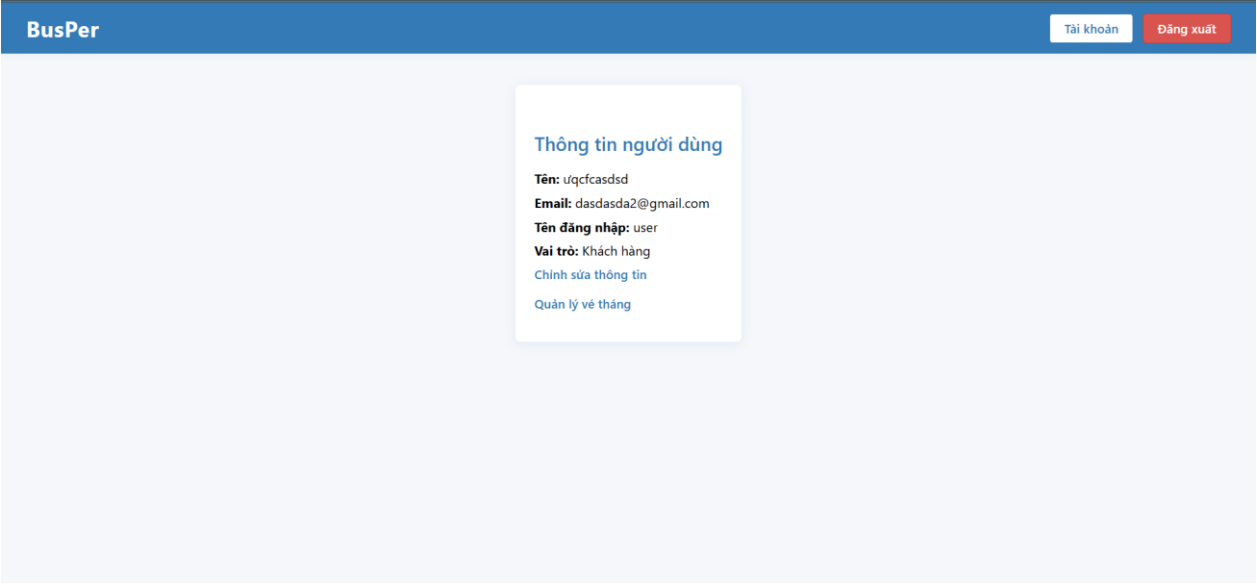
Các giao diện chính:

Giao diện trang chủ:



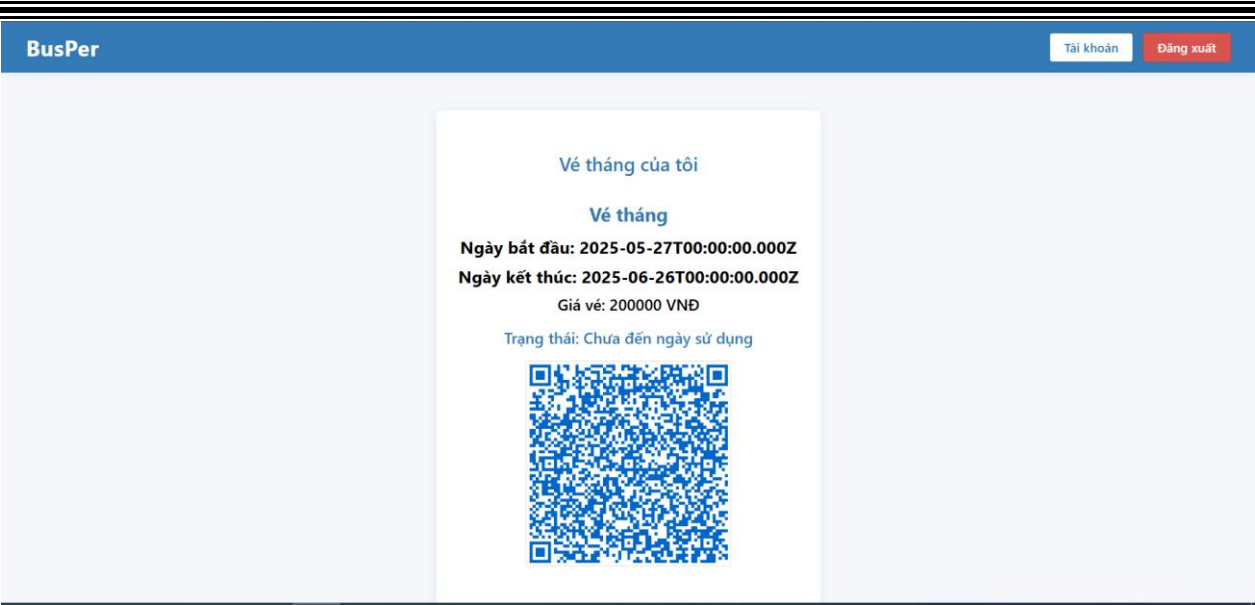
Hình 3.1: Hình ảnh giao diện trang chủ

Giao diện tài khoản:



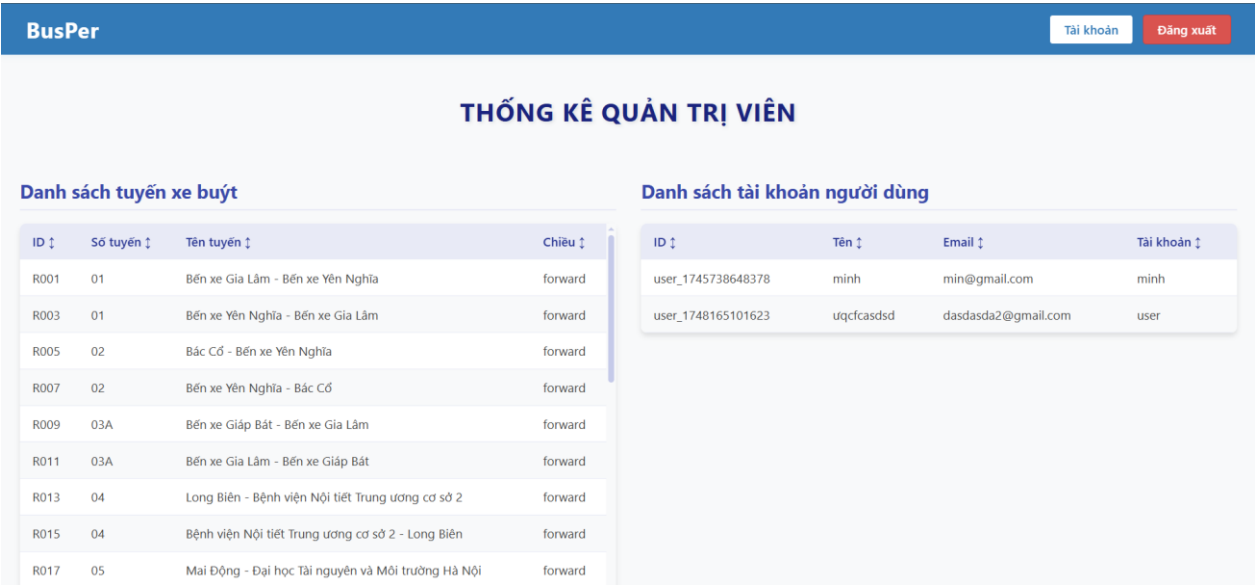
Hình 3.2: Hình ảnh giao diện tài khoản

Giao diện vé tháng:



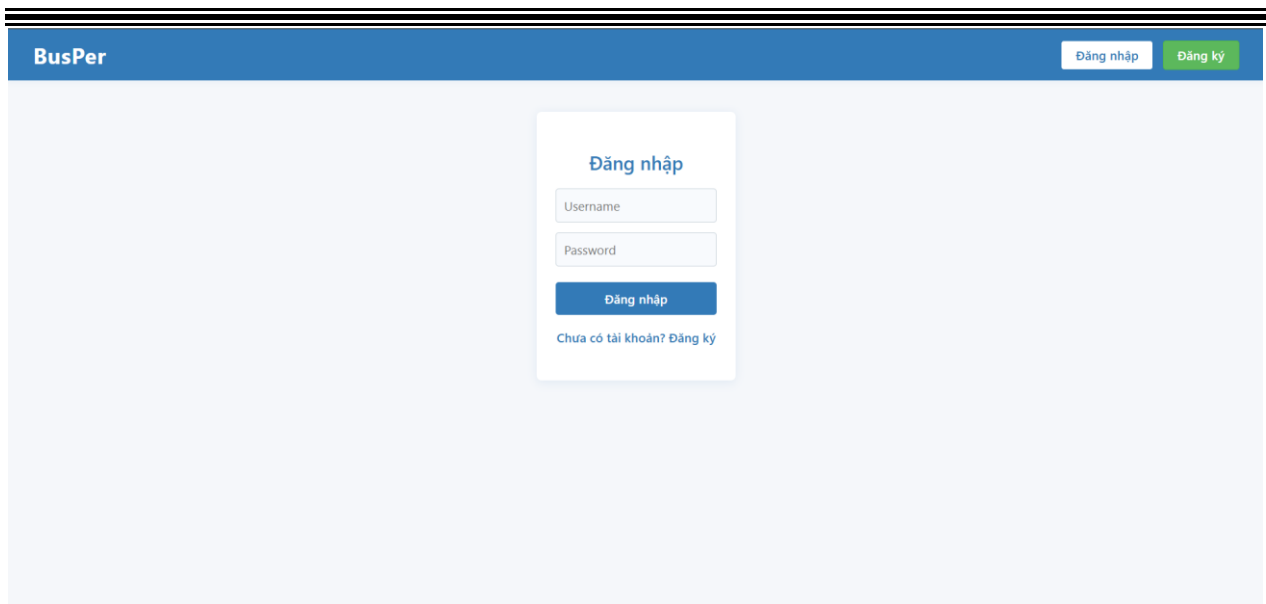
Hình 3.3: Hình ảnh giao diện vé tháng:

Giao diện thống kê:



Hình 3.4: Hình ảnh giao diện thống kê

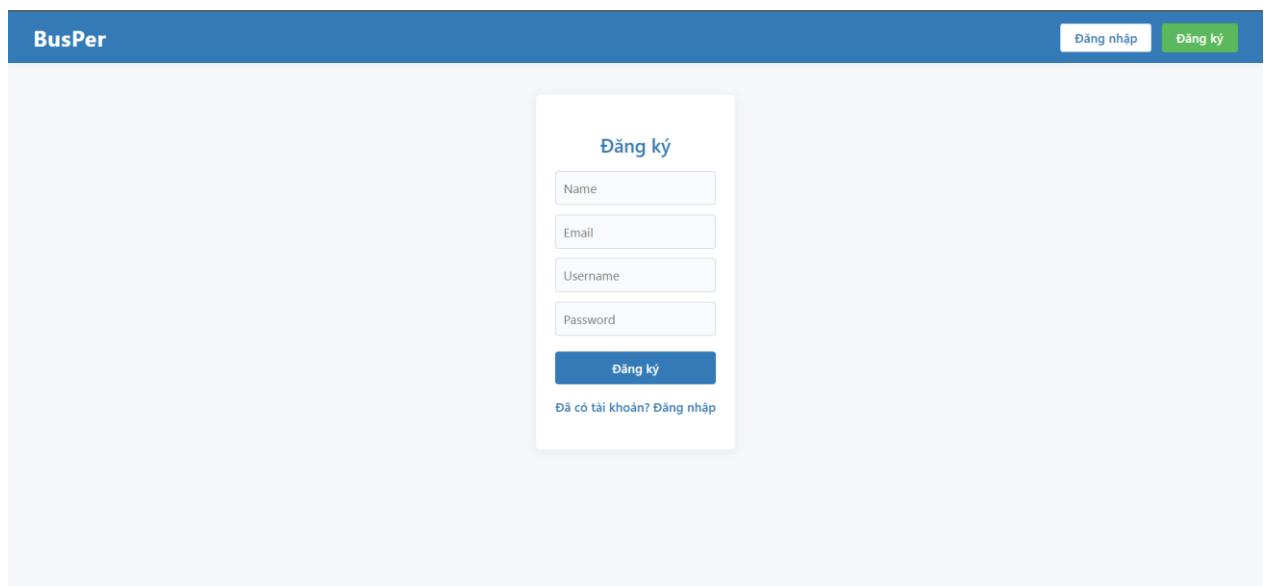
Giao diện đăng nhập:



The screenshot shows the login page of the BusPer application. At the top, there is a blue header bar with the 'BusPer' logo on the left and two buttons, 'Đăng nhập' (Login) and 'Đăng ký' (Register), on the right. The main content area is light blue and contains a white login form. The form has a title 'Đăng nhập' in blue. It includes two input fields: 'Username' and 'Password'. Below these fields is a blue button labeled 'Đăng nhập'. At the bottom of the form, there is a link that says 'Chưa có tài khoản? Đăng ký' (Don't have an account? Register).

Hình 3.5: Hình ảnh giao diện đăng nhập

Giao diện đăng ký:



The screenshot shows the registration page of the BusPer application. It has the same blue header bar as the login page, with the 'BusPer' logo and 'Đăng nhập' and 'Đăng ký' buttons. The main content area is light blue and contains a white registration form. The form has a title 'Đăng ký' in blue. It includes four input fields: 'Name', 'Email', 'Username', and 'Password'. Below these fields is a blue button labeled 'Đăng ký'. At the bottom of the form, there is a link that says 'Đã có tài khoản? Đăng nhập' (Already have an account? Login).

Hình 3.6: Hình ảnh giao diện đăng ký

3.3.2 Hạn chế

Mặc dù hệ thống quản lý xe buýt đã đạt được những kết quả tích cực, nhưng vẫn còn một số hạn chế cần cải thiện.

Thứ nhất, giao diện người dùng có thể cần được tối ưu hóa thêm để trở nên thân thiện và dễ sử dụng hơn, đặc biệt đối với người dùng không có nhiều kinh nghiệm về công nghệ.

Thứ hai, thuật toán tìm đường còn nhiều hạn chế chưa được tối ưu. Cần cải thiện thêm về thuật toán tìm đường

Thứ ba, còn một số tính năng không được hoàn thiện như khi tìm được các đoạn đường đi nhưng độ dài đường đi vẫn chưa được hiển thị, tính năng thống kê của quản lý vẫn còn sơ sài, cần cải thiện thêm nhiều chức năng hơn, chức năng mua vé chưa được thêm tính năng thanh toán mà mới chỉ dừng ở mức tạo ra mã QR

Thứ tư, cơ sở dữ liệu có thể chưa được tối ưu, nhiều dữ liệu có thể được suy ra từ dữ liệu khác trong cùng bảng hoặc các bảng có quan hệ đến bảng đó. Các dữ liệu được lưu cũng không chắc chắn đúng

3.3.3 Định hướng phát triển

Định hướng phát triển của trang web xe buýt có thể sẽ được sử dụng nhiều hơn trong tương lai để thúc đẩy người dùng sử dụng phương tiện công cộng, giảm thiểu ùn tắc.

Cải tiến giao diện người dùng. Tiếp tục tối ưu hóa giao diện để trở nên dễ sử dụng hơn, đặc biệt là đối với các khách hàng không quen với công nghệ. Đảm bảo hệ thống giao diện linh hoạt, dễ dàng tương thích với nhiều thiết bị và nền tảng.

Cần cải thiện thuật toán tìm đường để được tối ưu hơn (sử dụng các thuật toán có độ khó cao hơn). Có thể xây dựng cơ sở dữ liệu cho bản đồ riêng thay vì sử dụng Nominaton sẽ giúp tốc độ xử lý tốt hơn.

Hoàn thiện các tính năng còn đang dang dở: Tính năng tính khoảng cách cho các đường đi được tìm thấy cần được thêm vào để người dùng lựa chọn tuyến đường đi ngắn nhất. Tính năng mua vé tháng cần tích hợp thêm VNPay hoặc các chức năng thanh toán. Tính năng thống kê của quản lý cần được làm đẹp hơn (thêm chức năng thêm, sửa, xóa) và các chức năng khác hỗ trợ quản lý theo dõi trang web.