BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỒ ÁN TỐT NGHIỆP

XÂY DỰNG HỆ THỐNG VẬN TẢI HÀNH KHÁCH GHÉP TUYẾN ĐA NỀN TẢNG

Ngành: Công nghệ thông tin

Mã số: 748.02.01

Sinh viên thực hiện:

Phan Văn Tuấn

Lớp: CT5B

Người hướng dẫn:

ThS. Thái Thị Thanh Vân

Khoa công nghệ thông tin – Học viện Kỹ thuật mật mã

Hà Nội, 2025

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC HÌNH ẢNH

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn sâu sắc đến Ban lãnh đạo Học viện cùng các thầy cô Khoa Công nghệ Thông tin – Học viện Kỹ thuật Mật mã đã tạo điều kiện thuận lợi về cơ sở vật chất và cung cấp kiến thức chuyên môn quý báu trong suốt quá trình thực hiện đề tài. Em xin chân thành cảm ơn ThS. Thái Thị Thanh Vân, người hướng dẫn tận tâm, đã dành thời gian và tâm huyết để hướng dẫn, góp ý và định hướng nghiên cứu, triển khai đề tài. Sự chỉ dẫn của cô đã giúp em vượt qua những khó khăn và hoàn thành công việc một cách tốt nhất. Cuối cùng, em xin gửi lời cảm ơn đến gia đình và bạn bè đã luôn động viên, khích lệ và hỗ trợ tinh thần trong suốt quá trình thực hiện đồ án tốt nghiệp.

Hà Nội, Ngày Tháng Năm 2025

LỜI NÓI ĐẦU

Trong bối cảnh kinh tế và xã hội hiện đại, nhu cầu vận tải hành khách không ngừng gia tăng, đặc biệt tại các khu vực đô thị và vùng kinh tế trọng điểm. Tính đa dạng và phức tạp trong hành trình của người dân đòi hỏi các hệ thống vận tải phải linh hoạt, hiệu quả và có khả năng kết nối nhiều tuyến đường, nền tảng khác nhau. Tuy nhiên, nhiều hệ thống vận tải hiện nay vẫn tồn tại những hạn chế, khó khăn trong việc ghép tuyến, và chưa tối ưu các nhu cầu vận chuyển của người dân. Cụ thể, việc đặt xe ghép truyền thống thường dựa trên các phương thức thủ công như gọi điện, nhắn tin hoặc thỏa thuận trực tiếp giữa khách hàng, tài xế và đơn vị quản lý. Điều này dẫn đến nhiều hạn chế như thiếu minh bạch về giá cả, khó khăn trong việc tìm kiếm tuyến đường phù hợp, mất thời gian trong việc phân bổ tài xế và xử lý đặt tuyến. Hơn nữa, khách hàng thường gặp khó khăn trong việc tiếp cận thông tin về các tuyến xe ghép, lịch trình và tình trạng chỗ ngồi.

Trong xu thế chuyển đổi số và sự phát triển mạnh mẽ của công nghệ, việc xây dựng một hệ thống vận tải hành khách thông minh, đa nền tảng, với khả năng ghép tuyến linh hoạt không chỉ giải quyết bài toán hiện tại mà còn mở ra hướng đi bền vững cho ngành giao thông vận tải. Điều này không chỉ giúp giảm tải áp lực cho hệ thống hạ tầng giao thông mà còn cải thiện trải nghiệm người dùng, nâng cao chất lượng dịch vụ, đồng thời thúc đẩy sự phát triển kinh tế - xã hội.

Xuất phát từ thực tiễn nhu cầu vận tải hành khách ngày càng gia tăng và những hạn chế còn tồn tại trong mô hình đặt xe ghép tuyến truyền thống, đề tài này được lựa chọn nhằm nghiên cứu, phân tích, thiết kế và xây dựng một hệ thống đặt xe ghép tuyến thông minh, linh hoạt và đa nền tảng. Mục tiêu chính là số hóa quy trình vận hành cũ, tối ưu hóa trải nghiệm người dùng và nâng cao hiệu quả kết nối giữa hành khách và tài xế.

Báo cáo trình bày toàn bộ quá trình khảo sát, đánh giá hệ thống hiện tại, từ đó đề xuất mô hình và triển khai giải pháp kỹ thuật phù hợp. Cấu trúc báo cáo gồm các chương chính:

- Chương 1. Tổng quan về đề tài và các công nghệ sử dụng: Trình bày bối cảnh, lý do chọn đề tài và tổng quan về các công nghệ được áp dụng trong quá trình xây dựng hệ thống.
- Chương 2. Phân tích thiết kế hệ thống: Phân tích yêu cầu, xác định các chức năng chính và mô hình kiến trúc tổng thể của hệ thống.

- Chương 3. Triển khai, kiểm thử và đánh giá hệ thống: Trình bày quy trình hiện thực hóa hệ thống, các công cụ sử dụng trong quá trình triển khai, kết quả kiểm thử và đánh giá hiệu quả.

Sau thời gian thực hiện, hệ thống đã cơ bản đạt được các mục tiêu đề ra, mang lại một nền tảng trực tuyến tiện lợi và hiệu quả cho cả hành khách và nhà xe. Tuy nhiên, do hạn chế về thời gian và kinh nghiệm, đề tài không tránh khỏi những thiếu sót. Với tinh thần cầu thị, tôi rất mong nhận được sự cảm thông và những ý kiến đóng góp quý báu từ quý thầy cô, các bạn học viên và những người quan tâm để hệ thống này được hoàn thiện hơn. Hy vọng rằng, báo cáo này không chỉ là kết quả của quá trình nghiên cứu cá nhân mà còn đóng góp một phần nhỏ vào sự phát triển của các giải pháp công nghệ trong ngành vận tải hành khách tại Việt Nam, hướng tới một tương lai hiện đại và bền vững hơn.

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI VÀ CÁC CÔNG NGHỆ SỬ DỤNG

1.1. Tổng quan đề tài

1.1.1. Đặt vấn đề

Trong bối cảnh công nghệ số phát triển mạnh mẽ, nhu cầu di chuyển của con người ngày càng tăng cao, đặc biệt là tại các đô thị lớn và các khu vực có mật độ dân cư cao. Các dịch vụ vận tải hành khách theo tuyến cố định đóng vai trò quan trọng trong việc đáp ứng nhu cầu này. Tuy nhiên, việc đặt xe ghép truyền thống hiện nay chủ yếu dựa vào các phương thức thủ công như gọi điện thoại, nhắn tin hoặc thỏa thuận trực tiếp giữa khách hàng, tài xế và đơn vị quản lý. Điều này dẫn đến nhiều bất cập và hạn chế, ảnh hưởng đến hiệu quả hoạt động và trải nghiệm của người dùng. Cụ thể, việc thiếu vắng các nền tảng công nghệ trong quy trình đặt xe ghép gây ra những vấn đề sau:

- **Thiếu minh bạch về giá cả:** Khách hàng thường gặp khó khăn trong việc nắm bắt thông tin chính xác và rõ ràng về giá vé, các khoản phụ phí, dẫn đến tình trạng bị động trong quá trình giao dịch.
- **Khó khăn trong tìm kiếm tuyến đường phù hợp:** Việc tìm kiếm và lựa chọn tuyến đường, giờ khởi hành phù hợp tốn nhiều thời gian và công sức, đặc biệt khi có nhiều lựa chọn khác nhau.
- **Tốn thời gian phân bổ tài xế và xử lý chuyến xe:** Các đơn vị quản lý vận tải gặp khó khăn trong việc điều phối tài xế, sắp xếp lịch trình và xử lý các yêu cầu đặt xe, dẫn đến chậm trễ và giảm hiệu quả hoạt động.
- **Khả năng tiếp cận thông tin hạn chế:** Khách hàng khó tiếp cận thông tin về các tuyến xe ghép, lịch trình, tình trạng chỗ ngồi và các thông tin liên quan khác một cách nhanh chóng và thuận tiện.

Những hạn chế này không chỉ gây bất tiện cho người dùng mà còn làm giảm hiệu quả kinh doanh của các đơn vị vận tải, tăng chi phí hoạt động và giảm khả năng cạnh tranh. Do đó, việc phát triển một hệ thống đặt xe ghép tuyến đa nền tảng, ứng dụng công nghệ để số hóa và tự động hóa quy trình truyền thống, là một yêu cầu cấp thiết nhằm nâng cao hiệu quả, tối ưu hóa chi phí và cải thiện trải nghiệm người dùng.

1.1.2. Giới thiệu đề tài

Đề tài "Hệ thống đặt xe ghép tuyến đa nền tảng" tập trung vào việc nghiên cứu, thiết kế và xây dựng một ứng dụng nhằm số hóa quy trình đặt xe ghép truyền thống, từ đó giải quyết các vấn đề đã nêu trong phần "Đặt vấn đề". Mục tiêu chính là cung cấp một giải pháp công nghệ toàn diện, nâng cao hiệu quả hoạt động và cải thiện trải nghiệm người dùng trong lĩnh vực vận tải hành khách theo tuyến cố định.

Hệ thống được phát triển với các chức năng chính sau:

- **Quản lý tuyến đường:** Cho phép quản trị viên (admin) thiết lập, cập nhật và quản lý thông tin về các tuyến đường, lịch trình, giá vé.
- **Quản lý tài xế:** Hỗ trợ quản trị viên phân bổ tài xế, theo dõi hoạt động và quản lý thông tin liên quan đến tài xế.
- **Đặt xe:** Cung cấp cho khách hàng khả năng đặt xe một cách dễ dàng và nhanh chóng thông qua giao diện trực quan trên các nền tảng web và ứng dụng di động.
- **Gợi ý tuyến đường:** Tự động gợi ý các tuyến đường phù hợp dựa trên thông tin điểm đi và điểm đến của khách hàng.
- **Thông báo và theo dõi chuyến xe:** Cung cấp thông tin chi tiết về chuyến xe, thông báo thời gian thực và hỗ trợ theo dõi hành trình.

Bằng việc ứng dụng công nghệ, đề tài hướng tới việc hiện đại hóa dịch vụ xe ghép, mang lại sự tiện lợi, minh bạch và tin cậy cho cả người dùng và các đơn vị vận tải.

1.1.3. Phạm vi đề tài

- **Phạm vi chức năng:** Hệ thống tập trung vào các tính năng cốt lõi như quản lý tuyến đường, phân bổ tài xế, đặt xe, gợi ý tuyến đường, thông báo và theo dõi chuyến xe. Các chức năng bổ sung như thanh toán trực tuyến hoặc tích hợp trí tuệ nhân tạo để tối ưu hóa tuyến đường sẽ được xem xét trong giai đoạn phát triển sau.
- **Phạm vi nền tảng:** Hệ thống được thiết kế để hoạt động trên các nền tảng web và ứng dụng di động (Android và iOS), đảm bảo khả năng tiếp cận đa dạng cho người dùng.

- **Phạm vi địa lý:** Đề tài tập trung vào việc triển khai hệ thống tại thị trường Việt Nam, nơi dịch vụ xe ghép tuyến có nhu cầu cao, đặc biệt ở các khu vực đô thị và liên tỉnh.
- **Phạm vi thời gian:** Đề tài được thực hiện trong khuôn khổ đồ án học thuật, với thời gian phát triển và thử nghiệm trong vòng 3 tháng.

1.1.4. Tính cấp thiết của đề tài

Trong những năm gần đây, dịch vụ xe ghép tuyến tại Việt Nam đã chứng kiến sự tăng trưởng đáng kể, trở thành một phương thức di chuyển phổ biến nhờ tính linh hoạt và chi phí hợp lý so với các lựa chọn vận tải truyền thống. Tuy nhiên, sự phát triển nhanh chóng này cũng bộc lộ nhiều hạn chế do thiếu các giải pháp công nghệ toàn diện, gây ảnh hưởng đến cả người dùng và các đơn vị cung cấp dịch vụ.

Cụ thể, các vấn đề sau đây cho thấy tính cấp thiết của việc phát triển một hệ thống đặt xe ghép tuyến đa nền tảng:

- **Trải nghiệm người dùng kém:**
  - Khách hàng thường gặp khó khăn trong việc tìm kiếm thông tin về các tuyến xe, lịch trình và giá cả.
  - Quy trình đặt xe thủ công (qua điện thoại, tin nhắn) gây mất thời gian và bất tiện.
  - Thiếu sự minh bạch và tin cậy trong giao dịch.
- **Hiệu quả hoạt động thấp:**
  - Các đơn vị vận tải gặp khó khăn trong việc quản lý và điều phối xe, dẫn đến lãng phí nguồn lực.
  - Khả năng tối ưu hóa tuyến đường và ghép chuyến còn hạn chế.
  - Khó khăn trong việc mở rộng quy mô và cạnh tranh.

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ, việc ứng dụng công nghệ để giải quyết những vấn đề trên không chỉ là một yêu cầu tất yếu mà còn là cơ hội để nâng cao chất lượng dịch vụ, thúc đẩy sự phát triển bền vững của ngành vận tải hành khách. Một hệ thống đặt xe ghép tuyến đa nền tảng có thể mang lại những lợi ích thiết thực như:

- **Đối với người dùng:**
  - Dễ dàng tìm kiếm, đặt xe và theo dõi hành trình.
  - Tiết kiệm thời gian và chi phí di chuyển.

- Trải nghiệm dịch vụ tiện lợi, minh bạch và an toàn.

- **Đối với đơn vị vận tải:**
  - Tối ưu hóa quy trình quản lý và điều hành.
  - Nâng cao hiệu quả sử dụng phương tiện và tài xế.
  - Mở rộng thị trường và tăng khả năng cạnh tranh.

Do đó, đề tài "Hệ thống đặt xe ghép tuyến đa nền tảng" là một nghiên cứu có tính cấp thiết, đáp ứng nhu cầu thực tiễn và góp phần vào sự phát triển của ngành vận tải hành khách tại Việt Nam.

1.1.5. Mục tiêu đề tài

- **Mục tiêu tổng quát:** Phát triển một hệ thống đặt xe ghép tuyến đa nền tảng nhằm số hóa quy trình đặt xe truyền thống, nâng cao hiệu quả quản lý và trải nghiệm người dùng.
- **Mục tiêu cụ thể:**
  - Xây dựng hệ thống cho phép quản trị viên thiết lập tuyến đường, phân bổ tài xế và quản lý tuyến xe.
  - Cung cấp giao diện thân thiện cho khách hàng để tìm kiếm, đặt xe và theo dõi trạng thái chuyến xe.
  - Tích hợp thuật toán gợi ý tuyến đường dựa trên điểm đi và điểm đến của khách hàng.
  - Đảm bảo hệ thống hoạt động ổn định trên các nền tảng web và di động.
  - Tăng tính minh bạch và tiện lợi trong quy trình đặt xe ghép.

1.1.6. Ý nghĩa đề tài

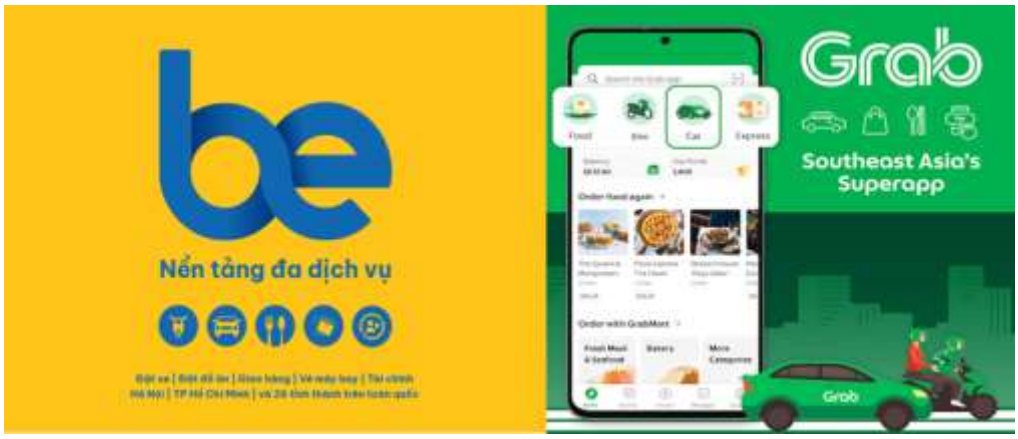
- **Ý nghĩa khoa học**: Đề tài góp phần nghiên cứu và ứng dụng các công nghệ hiện đại như phát triển ứng dụng đa nền tảng, cơ sở dữ liệu và thuật toán gợi ý trong lĩnh vực vận tải. Kết quả của đề tài có thể làm nền tảng cho các nghiên cứu tiếp theo về tối ưu hóa tuyến đường hoặc tích hợp trí tuệ nhân tạo.
- **Ý nghĩa thực tiễn**: Hệ thống giúp số hóa quy trình đặt xe ghép, giảm thiểu các thao tác thủ công, tăng cường tính minh bạch và cải thiện trải nghiệm của khách hàng. Đồng thời, hệ thống hỗ trợ các đơn vị cung cấp dịch vụ xe ghép nâng cao hiệu quả quản lý, giảm chi phí vận hành và tăng khả năng cạnh tranh trên thị trường.

- **Ý nghĩa xã hội**: Đề tài góp phần thúc đẩy chuyển đổi số trong lĩnh vực vận tải, nâng cao chất lượng dịch vụ và đáp ứng nhu cầu di chuyển ngày càng cao của người dân

1.2. Khảo sát các ứng dụng thực

Để hiểu rõ hơn về các giải pháp công nghệ hiện có trong lĩnh vực vận tải hành khách và xác định các yêu cầu cho hệ thống đặt xe ghép tuyến, đề tài đã tiến hành khảo sát một số ứng dụng thực tế, bao gồm:

1.2.1. Ứng dụng đặt xe công nghệ



Hình 1. Khảo sát ứng dụng Be, Grab.

- **Mô tả:** Grab và Be là các ứng dụng vận tải công nghệ phổ biến tại Việt Nam, cung cấp dịch vụ xe ôm và xe hơi theo yêu cầu. Các ứng dụng này sử dụng GPS để kết nối khách hàng với tài xế, hiển thị thông tin giá cả, lộ trình và thời gian di chuyển.
- **Phân tích:**
  - **Điểm mạnh:**
    + Giao diện thân thiện, dễ sử dụng.
    + Thanh toán trực tuyến tiện lợi.
    + Hệ thống đánh giá tài xế giúp nâng cao chất lượng dịch vụ.
  - **Điểm yếu:**
    + Chủ yếu phục vụ nhu cầu di chuyển cá nhân trong đô thị.
    + Không tập trung vào dịch vụ xe ghép tuyến liên tỉnh.
    + Thiếu tính năng gợi ý, lựa chọn hoặc quản lý lịch trình cố định.
- **Bài học rút ra:**
  - Cần xây dựng giao diện trực quan, dễ sử dụng.
  - Tích hợp GPS để theo dõi vị trí và tối ưu hóa lộ trình.

- Đảm bảo tính minh bạch trong thông tin cung cấp cho người dùng (giá cả, thời gian, lộ trình).

1.2.2. Ứng dụng đặt vé xe khách

- **Mô tả:** Nhà xe Văn Minh là một đơn vị vận tải hành khách cung cấp dịch vụ đặt vé xe khách qua website và ứng dụng. Ứng dụng cho phép khách hàng chọn tuyến đường, xem lịch trình và đặt vé trực tuyến.
- **Phân tích:**
  - **Điểm mạnh:**
    + Cung cấp thông tin chi tiết về các tuyến cố định và tình trạng chỗ ngồi.
    + Hỗ trợ đặt vé trực tuyến tiện lợi.
  - **Điểm yếu:**
    + Khả năng tương tác thời gian thực còn hạn chế.
    + Thiếu tính năng gợi ý tuyến đường linh hoạt hoặc hỗ trợ quản lý xe ghép.
    + Khó mở rộng do xe khách với giá cả lớn đi kèm các khoản thuế, phí.
- **Bài học rút ra:**
  - Cần tích hợp thuật toán gợi ý tuyến đường để tối ưu hóa việc ghép chuyến.
  - Tập trung vào các phương tiện vận tải loại vừa để tối ưu nhiều mặt.

1.2.3. Yêu cầu của hệ thống đặt xe

Dựa trên việc khảo sát các ứng dụng trên, hệ thống đặt xe ghép tuyến cần đáp ứng các yêu cầu sau:

- **Chức năng quản trị viên:**
  - Thiết lập và quản lý thông tin tuyến đường.
  - Quản lý thông tin tài xế và phân công tuyến đường.
  - Quản lý chuyến xe và theo dõi trạng thái theo thời gian thực.
- **Chức năng khách hàng:**
  - Tìm kiếm tuyến đường phù hợp dựa trên điểm đi/đến.
  - Đặt xe và thanh toán trực tuyến.
  - Xem thông tin chi tiết về lịch trình, giá cả và tình trạng chỗ ngồi.
  - Nhận thông báo và theo dõi trạng thái chuyến xe.

- **Chức năng tài xế:**
  - Nhận thông tin chi tiết về chuyến xe và lộ trình.
  - Tương tác với hệ thống qua ứng dụng di động.
- **Tính năng đặc thù:**
  - Thuật toán gợi ý tuyến đường dựa trên điểm đi/đến.
  - Tích hợp GPS để theo dõi lộ trình và hỗ trợ điều hướng.
  - Đảm bảo tính minh bạch trong giá cả và các thông tin liên quan.
- **Yêu cầu phi chức năng:**
  - Hiệu năng và độ tin cậy: Hệ thống phải hoạt động ổn định và xử lý yêu cầu nhanh chóng.
  - Bảo mật: Đảm bảo an toàn và bảo mật thông tin người dùng.

1.3. Các công cụ và công nghệ phát triển hệ thống

Phần này trình bày các công nghệ và công cụ được lựa chọn để phát triển hệ thống vận tải hành khách ghép tuyến đa nền tảng, đảm bảo hiệu suất, khả năng mở rộng và trải nghiệm người dùng tối ưu.

1.3.1. Ứng dụng di động:

- **React Native + TypeScript**: Framework mã nguồn mở được sử dụng để phát triển ứng dụng di động cho khách hàng (Rider App) và tài xế (Driver App). TypeScript tăng tính chặt chẽ về kiểu dữ liệu, hỗ trợ phát hiện lỗi sớm và nâng cao khả năng bảo trì.
- **Expo**: Công cụ hỗ trợ phát triển và xây dựng, cung cấp môi trường preview nhanh, hot-reload, và triển khai dễ dàng trên iOS/Android.
- **React Navigation v6 + Expo Router v3**: Quản lý điều hướng với cấu trúc stack, tab và file-based routing, đảm bảo tổ chức logic giao diện linh hoạt (xác thực, hồ sơ, lịch sử chuyến đi).
- **Zustand + Axios + React Hook Form + Zod**: Quản lý trạng thái toàn cục, giao tiếp API, xử lý biểu mẫu và xác thực dữ liệu đầu vào, tối ưu trải nghiệm người dùng và giảm lỗi logic.

- **NativeWind + Lucide React Native**: NativeWind tích hợp Tailwind CSS để định kiểu giao diện nhanh, kết hợp với Lucide Icons cung cấp biểu tượng hiện đại, nâng cao tính thẩm mỹ.
- **Date-fns**: Thư viện xử lý ngày giờ, hỗ trợ định dạng và so sánh thời gian hiệu quả.
- **Goong API/Google Maps API**: Tích hợp bản đồ và định vị, phục vụ gợi ý địa điểm, định tuyến và theo dõi hành trình thời gian thực.

1.3.2. Trang Website Quản lý

- **React + TypeScript**: Nền tảng phát triển Web Admin, đảm bảo giao diện tương tác cao, dễ kiểm thử và bảo trì lâu dài.
- **Tailwind CSS + Lucide React**: Tailwind CSS định kiểu linh hoạt bằng class utility, kết hợp Lucide Icons tạo giao diện nhất quán và hiện đại.
- **Zustand + React Context API + React Query**: Quản lý trạng thái và truy vấn dữ liệu bất đồng bộ, hỗ trợ caching và revalidation để tối ưu hiệu suất.
- **Framer Motion**: Áp dụng hiệu ứng chuyển đổi mượt mà (modal, dropdown), nâng cao trải nghiệm người dùng.

1.3.3. Backend Server

- **Node.js + TypeScript (Kiến trúc MVC)**: Xử lý nghiệp vụ phía server với hiệu suất cao, tổ chức theo mô hình Model-View-Controller để đảm bảo tách biệt và mở rộng.
- **MongoDB + Redis**: MongoDB lưu trữ dữ liệu chính (dạng document) với replica set trong Docker; Redis làm cache cho dữ liệu thời gian thực (vị trí, trạng thái chuyến đi).
- **JWT**: Sử dụng JSON Web Token cho xác thực và phân quyền, đảm bảo an toàn giao tiếp client-server.
- **RESTful API + Error handler**: Cung cấp API thống nhất, tích hợp Error handler để xử lý lỗi toàn cục và gắn token tự động.

1.3.4. Hạ tầng và Triển khai

- **Docker + Docker Compose**: Đóng gói và quản lý container cho backend, MongoDB, Redis, đảm bảo đồng nhất môi trường phát triển và sản xuất.

- **Vercel**: Triển khai Web Admin với CI/CD tự động từ GitHub, hỗ trợ preview pull request.
- **Google Cloud Platform (GCP)**: Hạ tầng đám mây cho server, cơ sở dữ liệu và dịch vụ Redis, đảm bảo độ tin cậy và khả năng mở rộng.
- **dotenv**: Quản lý biến môi trường (URL, secret key), tách biệt thông tin nhạy cảm khỏi mã nguồn.

1.3.5. Dịch vụ Hỗ trợ

- **Firebase Authentication**: Xác thực người dùng qua SMS OTP, hỗ trợ đăng ký và xác minh ban đầu.
- **Firebase Storage**: Lưu trữ tệp (ảnh giấy tờ, phương tiện), cung cấp URL an toàn cho hệ thống.
- **Firebase Cloud Messaging (FCM)**: Gửi thông báo đẩy cho ứng dụng di động (trạng thái chuyến đi, thông báo hệ thống).

1.3.6. Công cụ Phát triển và Hỗ trợ

- **Visual Studio Code (VS Code)**: Môi trường lập trình chính, tích hợp ESLint, Prettier và Tailwind IntelliSense để nâng cao chất lượng mã nguồn.
- **Git + GitHub**: Quản lý mã nguồn và hợp tác nhóm, hỗ trợ CI/CD.
- **Postman**: Kiểm thử API, mô phỏng yêu cầu và xác thực phản hồi.
- **MongoDB Compass**: Thao tác trực quan với MongoDB (xem document, tạo index).

Tổng kết chương 1

Chương 1 đã trình bày tổng quan về đề tài "Hệ thống đặt xe ghép tuyến đa nền tảng", bao gồm việc đặt vấn đề, giới thiệu đề tài, xác định phạm vi, nhấn mạnh tính cấp thiết, mục tiêu và ý nghĩa của đề tài. Phần khảo sát các ứng dụng thực tế đã phân tích các hệ thống như Grab, Be, và nhà xe Văn Minh, từ đó xác định các yêu cầu cụ thể cho hệ thống.

Các công nghệ sử dụng, bao gồm React Native, ReactJS, Node.js và MongoDB, được giới thiệu với vai trò cụ thể trong dự án, nhấn mạnh vào các công nghệ đặc thù như Zustand và JWT.

Nội dung chương này tạo nền tảng lý thuyết và thực tiễn cho việc phân tích, thiết kế và triển khai hệ thống trong các chương tiếp theo.

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2.1. Phân tích yêu cầu

2.1.1. Tác nhân hệ thống

Tác nhân là một thực thể bên ngoài (người hoặc hệ thống khác) tương tác với hệ thống. Trong hệ thống đặt xe ghép tuyến này, chúng ta có các tác nhân sau:

- **Người dùng chưa có tài khoản (Non-user):** Đây là những người chưa từng sử dụng hệ thống và chưa có tài khoản. Tác nhân này chủ yếu tương tác với hệ thống để thực hiện chức năng "Đăng ký" tài khoản.
- **Khách hàng (Rider):** Đây là những người dùng đã có tài khoản trong hệ thống và sử dụng các chức năng để đặt xe. Các chức năng chính mà khách hàng sử dụng bao gồm: Đăng nhập, Đặt cuốc xe, Hủy cuốc xe, Thanh toán, Đánh giá tài xế, Xem lịch sử, Cập nhật thông tin cá nhân, và Liên hệ hỗ trợ.
- **Tài xế (Driver):** Đây là những người dùng đã có tài khoản tài xế trong hệ thống và sử dụng các chức năng để quản lý các chuyến xe. Các chức năng chính mà tài xế sử dụng bao gồm: Đăng nhập, Nhận chuyến xe, Hoàn thành chuyến xe, và Nạp/rút tiền ví điện tử.
- **Quản trị viên (Admin):** Đây là người quản lý và vận hành hệ thống. Các chức năng chính mà quản trị viên sử dụng bao gồm: Quản lý người dùng, Cấu hình hệ thống, Xem báo cáo thống kê, và Trợ giúp và phản hồi.

2.1.2. Các yêu cầu chức năng

Yêu cầu chức năng mô tả những gì hệ thống phải thực hiện, và cần được diễn đạt một cách rõ ràng, ngắn gọn và có thể kiểm chứng được cho từng thành phần. Sau đây là các yêu cầu chức năng theo từng tác nhân:

Bảng 1. Các chức năng theo từng tác nhân

STT	Tác nhân	Chức năng
1	Người dùng chưa có tài khoản (Non-user)	• Đăng ký • Liên hệ hỗ trợ
2	Khách hàng	• Đăng nhập bằng số điện thoại.

	(Rider)	• Tìm kiếm và đặt xe phù hợp: chọn điểm đón, điểm đến, thời gian mong muốn. Thanh toán cho chuyến xe. • Theo dõi trạng thái chuyến đi theo thời gian thực (real-time). • Hủy chuyến nếu cần. • Xem lịch sử các chuyến đi đã đặt. • Nhận thông báo khi có khi xe đến nơi, hoàn tất chuyến,... • Đánh giá tài xế sau chuyến đi.
3	Tài xế (Driver)	• Đăng nhập bằng số điện thoại. • Gửi hồ sơ xác minh giấy tờ (ảnh giấy phép lái xe, đăng ký xe,...). • Nhận thông báo. • Xem lộ trình, bật theo dõi GPS. • Cập nhật trạng thái chuyến đi (đang đón, đang chở, đã hoàn tất). • Xem lịch sử chuyến đã thực hiện. • Quản lý thông tin cá nhân, phương tiện. • Nạp rút/tiền ví điện tử
4	Quản trị viên (Admin)	• Đăng nhập hệ thống quản trị. • Duyệt tài khoản tài xế và kiểm tra hồ sơ. • Tạo tuyến đường và phân công cho tài xế. • Quản lý tài khoản người dùng (chặn/mở khoá). • Quản lý cấu hình hệ thống: các loại giá, thuế, phí,... • Theo dõi thống kê, báo cáo: số lượng chuyến đi, tài xế đang hoạt động, phản hồi người dùng,...

2.1.3. Yêu cầu phi chức năng

Yêu cầu phi chức năng mô tả các thuộc tính chất lượng của hệ thống. Trong hệ thống đặt xe ghép tuyến này, chúng ta có các yêu cầu phi chức năng sau:

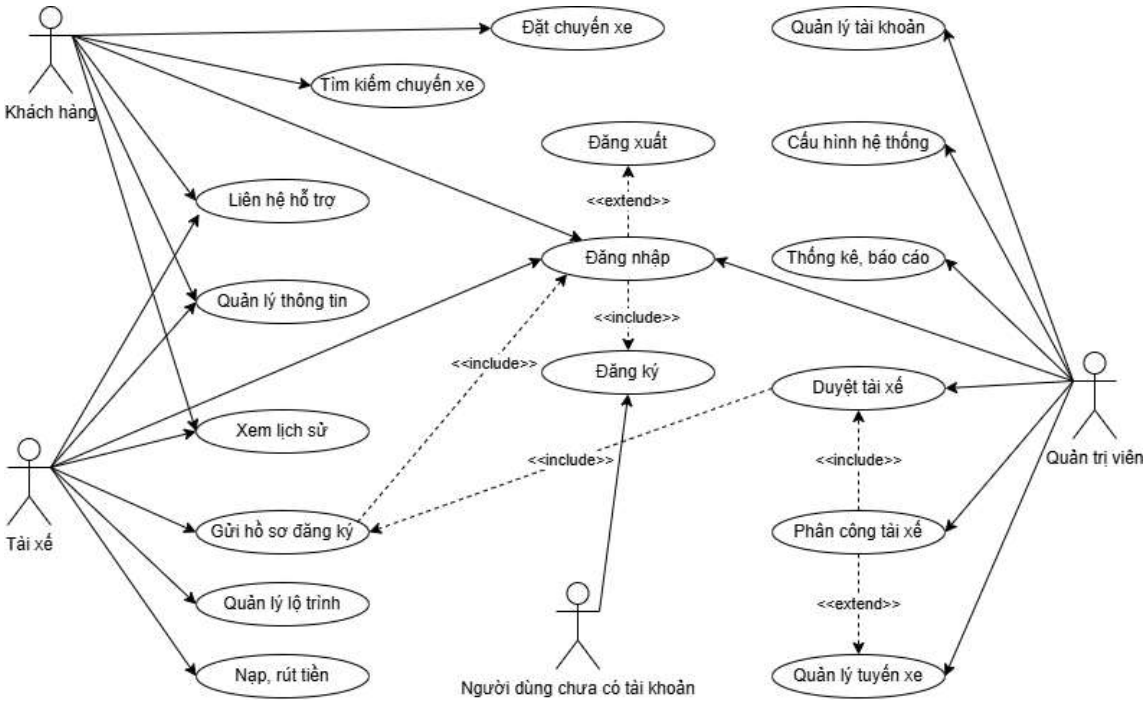
- **Khả năng mở rộng**: Hệ thống được thiết kế theo kiến trúc module, dễ dàng mở rộng thêm các tính năng trong tương lai (ví dụ: thanh toán điện tử, đặt chuyến nhiều chặng,...).

- **Hiệu năng**: Tối ưu phản hồi nhanh cho các thao tác chính như đặt xe, định vị, cập nhật trạng thái chuyến đi.
- **Bảo mật**: Hệ thống sử dụng xác thực JWT, dữ liệu nhạy cảm như token được lưu trữ an toàn. Các hình ảnh cá nhân (hồ sơ tài xế) được bảo vệ qua Firebase Storage + rule kiểm tra quyền truy cập.
- **Tính sẵn sàng**: Triển khai backend và cơ sở dữ liệu trên nền tảng đám mây, sử dụng Docker và replica set để đảm bảo hệ thống luôn hoạt động ổn định.
- **Khả năng tương thích đa nền tảng**: Ứng dụng di động hỗ trợ cả Android và iOS nhờ sử dụng React Native.
- **Dễ sử dụng**: Giao diện người dùng được thiết kế trực quan, đơn giản, tối ưu cho người dùng phổ thông.

2.2. Mô hình hóa yêu cầu

2.2.1. Biểu đồ usecase tổng quát

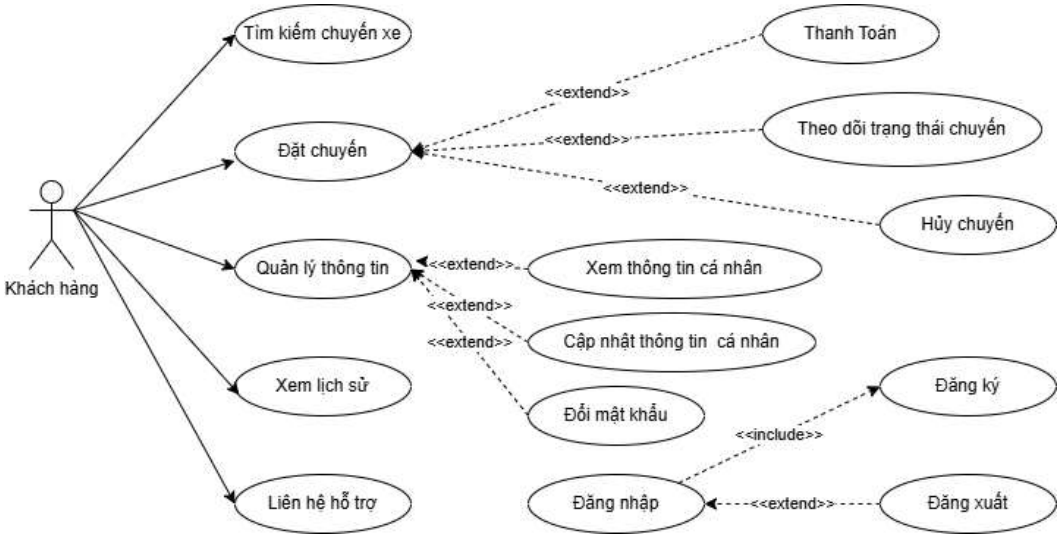
Sơ đồ Use Case tổng quát mô tả các chức năng chính của hệ thống và mối quan hệ giữa hệ thống với ba tác nhân chính: Khách hàng, Tài xế và Quản trị viên. Mỗi tác nhân có quyền truy cập vào các chức năng khác nhau tùy theo vai trò của họ trong hệ thống. Hệ thống yêu cầu người dùng phải **đăng nhập** để thực hiện hầu hết các chức năng như: đặt chuyến, gửi hồ sơ đăng ký, quản lý thông tin, phân công tài xế... Trong trường hợp chưa có tài khoản, người dùng cần thực hiện **đăng ký** trước khi đăng nhập. Các mối quan hệ “include” và “extend” được sử dụng để thể hiện các ràng buộc và mở rộng luồng nghiệp vụ phù hợp với logic hoạt động của hệ thống.



Hình 2. Biểu đồ usecase tổng quát

2.2.2. Biểu đồ usecase của tác nhân khách hàng

Sơ đồ use case mô tả các chức năng chính mà người dùng với vai trò **Khách hàng** có thể thực hiện trong hệ thống. Các chức năng bao gồm tìm kiếm và đặt chuyến, theo dõi trạng thái chuyến đi, thanh toán, quản lý thông tin cá nhân, xem lịch sử và liên hệ hỗ trợ. Hầu hết các hành động đều yêu cầu người dùng **đăng nhập**, và có thể được mở rộng tùy theo ngữ cảnh thực hiện như thanh toán hoặc hủy chuyến. Các chức năng như đăng ký tài khoản được bao gồm trong luồng đăng nhập dành cho người dùng chưa có tài khoản.

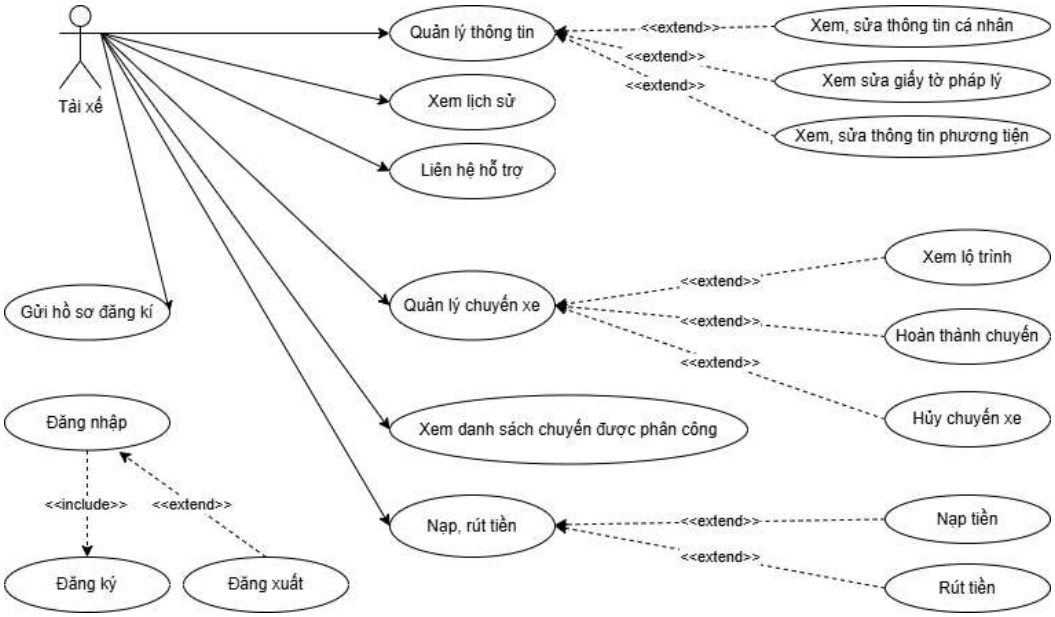


Hình 3. Biểu đồ usecase của tác nhân khách hàng

2.2.3. Biểu đồ usecase của tác nhân tài xế

Sơ đồ thể hiện các chức năng chính mà tài xế có thể thực hiện trong hệ thống. Mọi chức năng đều yêu cầu tài xế **đăng nhập trước**, trong đó quá trình đăng ký tài khoản là điều kiện tiên quyết nếu chưa có tài khoản.

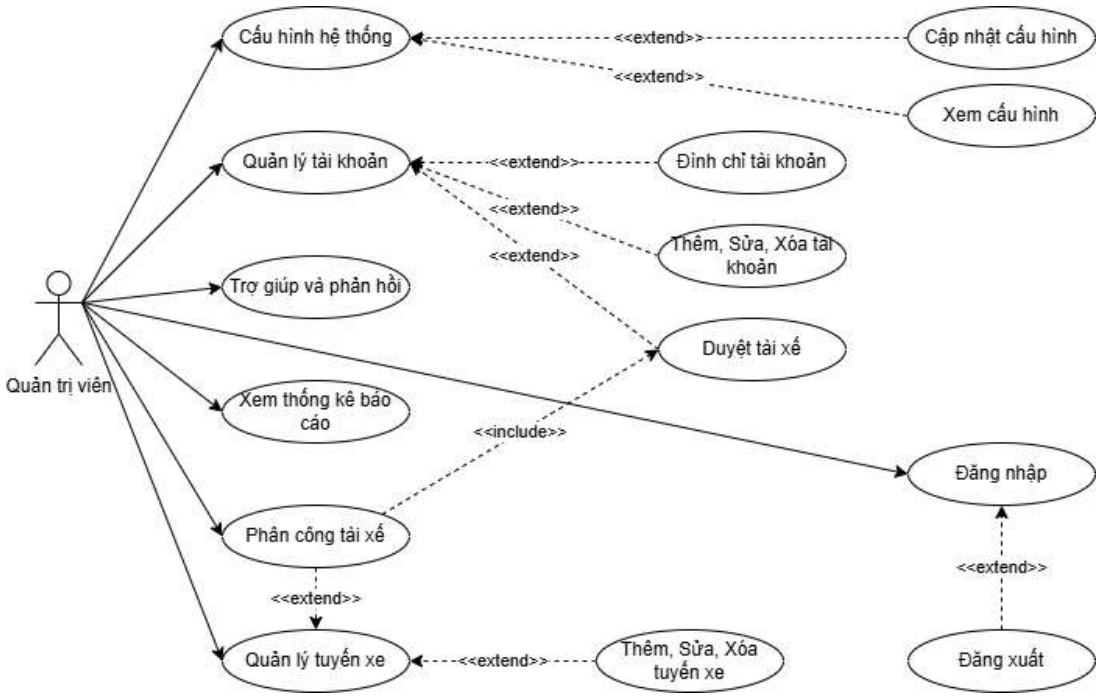
Sau khi **tài khoản tài xế được duyệt** bởi quản trị viên, tài xế có thể thực hiện các thao tác như quản lý chuyến xe, cập nhật thông tin cá nhân, nạp/rút tiền, gửi hồ sơ đăng ký, hoặc xem lịch sử hoạt động. Một số chức năng như hoàn thành chuyến, hủy chuyến, hay xem lộ trình chỉ xuất hiện khi có chuyến xe đã được hệ thống phân công.



Hình 4. Biểu đồ usecase của tác nhân tài xế

2.2.4. Biểu đồ usecase của tác nhân quản trị viên

Sơ đồ thể hiện các chức năng chính mà Quản trị viên có thể thực hiện trong hệ thống. Các chức năng bao gồm quản lý tài khoản người dùng, duyệt và phân công tài xế, cấu hình các thông số hệ thống như biểu phí, cũng như theo dõi thống kê, báo cáo. Quản trị viên phải **đăng nhập trước** để thực hiện các chức năng quản trị (Tài khoản quản trị viên **không thể đăng ký** mà do hệ thống cấp. Một số chức năng như xử lý phản hồi hoặc cấu hình chi tiết hệ thống có thể là các phần mở rộng được thực hiện tùy theo nhu cầu.



Hình 5. Biểu đồ usecase của tác nhân quản trị viên

2.3. Đặc tả ca sử dụng

Bảng 2. Đặc tả chức năng "Đăng ký"

Usecase	Đăng ký
Tác nhân	Người dùng chưa có tài khoản (Non-user)
Mô tả	Cho phép người dùng đăng ký tài khoản mới
Điều kiện	1. Người dùng chưa có tài khoản trong hệ thống. 2. Thiết bị của người dùng có kết nối internet.
Luồng chính	1. Người dùng mở ứng dụng và chọn chức năng "Đăng ký". 2. Hệ thống hiển thị màn hình đăng ký. 3. Người dùng nhập số điện thoại. 4. Hệ thống kiểm tra định dạng số điện thoại. 5. Hệ thống gửi mã xác thực (OTP) đến số điện thoại. 6. Người dùng nhập mã xác thực (OTP). 7. Hệ thống xác thực mã OTP. 8. Người dùng nhập các thông tin cá nhân (Họ tên, Email, Mật khẩu).

	9. Hệ thống kiểm tra tính hợp lệ của thông tin (định dạng email, độ mạnh của mật khẩu). 10. Người dùng xác nhận thông tin và hoàn tất đăng ký. 11. Hệ thống tạo tài khoản mới.
Luồng phụ	1. Số điện thoại đã tồn tại -> Hệ thống thông báo lỗi và yêu cầu nhập số điện thoại khác. 2. Mã xác thực (OTP) không hợp lệ hoặc hết hạn -> Hệ thống yêu cầu người dùng nhập lại hoặc gửi lại mã. 3. Thông tin cá nhân không hợp lệ (email, mật khẩu) -> Hệ thống thông báo lỗi và yêu cầu nhập lại. 4. Mất kết nối internet -> Hệ thống thông báo lỗi và yêu cầu kiểm tra kết nối.
Kết quả	Tài khoản người dùng được tạo thành công. Hệ thống chuyển người dùng đến màn hình đăng nhập.

Bảng 3. Đặc tả chức năng “Đăng nhập”

Usecase	Đăng nhập
Tác nhân	Người dùng đã có tài khoản (User)
Mô tả	Cho phép người dùng đăng nhập vào hệ thống
Điều kiện	1. Người dùng đã có tài khoản hợp lệ trong hệ thống. 2. Thiết bị của người dùng có kết nối internet.
Luồng chính	1. Người dùng mở ứng dụng và chọn chức năng "Đăng nhập". 2. Hệ thống hiển thị màn hình đăng nhập. 3. Người dùng nhập số điện thoại. 4. Hệ thống kiểm tra định dạng số điện thoại. 5. Người dùng nhập mật khẩu. 6. Hệ thống kiểm tra tính hợp lệ của mật khẩu. 7. Hệ thống xác thực số điện thoại và mật khẩu với cơ sở dữ liệu. 8. Hệ thống tạo phiên đăng nhập (token) và trả về dữ liệu. 9. Hệ thống chuyển người dùng đến trang chủ/màn hình chính.

Luồng phụ	1. Số điện thoại không tồn tại hoặc mật khẩu không đúng -> Hệ thống thông báo lỗi và yêu cầu nhập lại. 2. Tài khoản bị khóa -> Hệ thống thông báo tài khoản bị khóa (nếu có chức năng này). 3. Mất kết nối internet -> Hệ thống thông báo lỗi và yêu cầu kiểm tra kết nối.
Kết quả	1. Người dùng được đăng nhập thành công vào hệ thống. 2. Hệ thống chuyển người dùng đến trang chủ/màn hình chính. 3. Token đăng nhập được lưu trữ.

Bảng 4. Đặc tả chức năng “Đặt cuốc xe”

Usecase	Đặt cuốc xe
Tác nhân	Khách hàng
Mô tả	Cho phép khách hàng tìm kiếm lộ trình và đặt cuốc xe
Điều kiện	1. Người dùng đã đăng nhập. 2. Hệ thống có sẵn các tuyến xe và tài xế hoạt động. 3. Hệ thống có dữ liệu về các lộ trình (tuyến đường, điểm đón/trả khách). 4. Thiết bị của người dùng có kết nối internet.
Luồng chính	1. Khách hàng mở ứng dụng/website và chọn chức năng "Đặt xe". 2. Hệ thống hiển thị màn hình đặt xe. 3. Khách hàng nhập/chọn điểm đón. 4. Khách hàng nhập/chọn điểm đến. 5. Hệ thống gợi ý các địa điểm (nếu có). 6. Hệ thống tìm kiếm và hiển thị danh sách các lộ trình phù hợp (bao gồm thời gian di chuyển dự kiến, giá cả, v.v.). 7. Khách hàng chọn một lộ trình từ danh sách. 8. Hệ thống hiển thị thông tin chi tiết về lộ trình (nếu có) và các tùy chọn khác (loại xe). 9. Khách hàng chọn loại xe (nếu có tùy chọn). 10. Khách hàng xem xét và xác nhận thông tin chuyến xe.

	11. Hệ thống gửi yêu cầu đặt xe đến hệ thống. 12. Hệ thống xác nhận đặt xe thành công và hiển thị thông tin chi tiết về chuyến xe (mã chuyến, thông tin tài xế - nếu có, v.v.).
Luồng phụ	1. Không tìm thấy lộ trình phù hợp -> Hệ thống thông báo và đề xuất các lựa chọn khác (thời gian, ngày, v.v.). 2. Địa điểm nhập không hợp lệ -> Hệ thống yêu cầu nhập lại. 3. Người dùng hủy tìm kiếm/đặt xe ở bất kỳ bước nào -> Hệ thống quay lại màn hình trước đó. 4. Mất kết nối internet -> Hệ thống thông báo lỗi và yêu cầu kiểm tra kết nối.
Kết quả	1. Thông tin chuyến xe được lưu vào hệ thống. 2. Thông báo đặt xe thành công được hiển thị cho người dùng. 3. Người dùng có thể theo dõi thông tin chi tiết về chuyến xe đã đặt.

Bảng 5. Đặc tả chức năng “Theo dõi chuyến đi”

Usecase	Theo dõi chuyến đi
Tác nhân	Khách hàng
Mô tả	Cho phép khách hàng theo dõi thông tin chi tiết và trạng thái của cuốc xe đã đặt.
Điều kiện	1. Khách hàng đã đăng nhập. 2. Khách hàng đã đặt cuốc xe thành công.
Luồng chính	1. Khách hàng chọn chức năng "Xem trạng thái chuyến đi". 2. Hệ thống hiển thị thông tin chi tiết về chuyến đi: a. Điểm đón và điểm đến. b. Thời gian dự kiến đón/đến. c. Thông tin tài xế. d. Vị trí tài xế trên bản đồ (nếu có). e. Trạng thái chuyến đi. f. Giá cước ước tính/thực tế. g. Phương thức thanh toán.

	3. Hệ thống cập nhật thông tin chuyến đi theo thời gian thực (nếu có).
Luồng phụ	1. **Mất kết nối internet:** a. Hệ thống hiển thị thông báo lỗi. b. Hệ thống hiển thị thông tin cuối cùng đã tải được (nếu có). 2. **Không có cuốc xe đang diễn ra/sắp diễn ra:** a. Hệ thống hiển thị thông báo không có chuyến đi nào để theo dõi. 3. **Lỗi hệ thống:** a. Hệ thống hiển thị thông báo lỗi chung.
Kết quả	1. Khách hàng có thể theo dõi thông tin chi tiết và trạng thái của cuốc xe đã đặt. 2. Khách hàng được cập nhật về vị trí tài xế và thời gian di chuyển. 3. Khách hàng có thể nắm bắt được tiến trình của chuyến đi.

Bảng 6. Đặc tả chức năng "Hủy cuốc xe"

Usecase	Hủy cuốc xe
Tác nhân	Khách hàng
Mô tả	Cho phép khách hàng hủy cuốc xe đã đặt
Điều kiện	1. Cuốc xe đã được đặt thành công. 2. Thời gian hủy còn cho phép theo quy định của hệ thống. 3. Lý do hủy hợp lệ (nếu có yêu cầu).
Luồng chính	1. Khách hàng chọn cuốc xe cần hủy trên ứng dụng/website. 2. Hệ thống hiển thị màn hình xác nhận hủy. 3. Khách hàng chọn lý do hủy từ danh sách hoặc nhập lý do (nếu có). 4. Khách hàng xác nhận hủy. 5. Hệ thống kiểm tra điều kiện hủy (thời gian, lý do). 6. Hệ thống thông báo hủy thành công. 7. Hệ thống gửi thông báo về việc hủy chuyến đến tài xế (nếu có).

Luồng phụ	1. Thời gian hủy không hợp lệ (ví dụ: quá gần giờ khởi hành) -> Hệ thống thông báo và từ chối hủy. 2. Khách hàng không xác nhận hủy -> Hệ thống quay lại màn hình chi tiết cuốc xe. 3. Hủy không thành công do lỗi hệ thống (ví dụ: mất kết nối) -> Hệ thống thông báo lỗi và hướng dẫn khách hàng.
Kết quả	1. Cuốc xe bị hủy khỏi hệ thống. 2. Thông tin hủy chuyến được lưu lại. 3. Khách hàng và tài xế (nếu có) nhận được thông báo về việc hủy chuyến. 4. Hệ thống cập nhật trạng thái cuốc xe và có thể điều phối lại tài xế (nếu cần).

Bảng 7. Đặc tả chức năng "Thanh toán"

Usecase	Thanh toán
Tác nhân	Khách hàng
Mô tả	Cho phép khách hàng thực hiện thanh toán cho cuốc xe đã hoàn thành.
Điều kiện	1. Cuốc xe đã hoàn thành và hệ thống đã tính toán được số tiền cần thanh toán. 2. Khách hàng đã chọn phương thức thanh toán hợp lệ (nếu cần).
Luồng chính	1. Hệ thống hiển thị thông tin chi tiết về cuốc xe. 2. Hệ thống hiển thị các phương thức thanh toán khả dụng. 3. Khách hàng chọn phương thức thanh toán. 4. **Nếu chọn tiền mặt:** Hệ thống hiển thị thông báo thanh toán trực tiếp cho tài xế. 5. **Nếu chọn thẻ tín dụng/ví điện tử:** a. Hệ thống chuyển đến màn hình nhập thông tin thẻ/ví (nếu cần). b. Khách hàng nhập thông tin thẻ/ví. c. Hệ thống xác thực thông tin thẻ/ví. d. Hệ thống gửi yêu cầu thanh toán đến cổng thanh toán.

	e. Cổng thanh toán xử lý giao dịch. f. Hệ thống nhận kết quả từ cổng thanh toán. 6. Hệ thống hiển thị thông báo thanh toán thành công. 7. Hệ thống cập nhật trạng thái cuốc xe thành "Đã thanh toán".
Luồng phụ	1. **Thông tin thẻ/ví không hợp lệ:** Hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại. 2. **Thanh toán không thành công:** Hệ thống hiển thị thông báo lỗi và yêu cầu thử lại hoặc chọn phương thức khác. 3. **Mất kết nối internet:** Hệ thống hiển thị thông báo lỗi và hướng dẫn khách hàng. 4. **Khách hàng hủy thanh toán:** Hệ thống quay lại màn hình chi tiết cuốc xe hoặc chọn phương thức.
Kết quả	1. **Thanh toán thành công:** Hệ thống cập nhật trạng thái cuốc xe và tạo biên lai (nếu có). 2. **Thanh toán không thành công:** Hệ thống thông báo lỗi cho khách hàng.

Bảng 8. Đặc tả chức năng “Xem lịch sử chuyến đi”

Usecase	Xem lịch sử chuyến đi
Tác nhân	Khách hàng, Tài xế
Mô tả	Cho phép người dùng (Khách hàng hoặc Tài xế) có thể xem lại lịch sử các chuyến đi đã thực hiện, bao gồm thông tin chi tiết về từng chuyến.
Điều kiện	Người dùng đã đăng nhập.
Luồng chính	1. Người dùng chọn chức năng "Xem lịch sử chuyến đi". 2. Hệ thống hiển thị danh sách các chuyến đi đã thực hiện, sắp xếp theo thứ tự thời gian (có thể hiển thị các thông tin tóm tắt như ngày giờ, điểm đón/trả, giá cước). 3. Người dùng chọn một chuyến đi cụ thể để xem thông tin chi tiết.

	4. Hệ thống hiển thị thông tin chi tiết của chuyến đi được chọn (điểm đón/trả, thời gian, quãng đường, giá cước, thông tin tài xế/khách hàng, trạng thái chuyến đi).
Luồng phụ	1. **Không có lịch sử chuyến đi:** Hệ thống hiển thị thông báo "Không có lịch sử chuyến đi". 2. **Lỗi kết nối:** Nếu có lỗi kết nối trong quá trình tải dữ liệu, hệ thống hiển thị thông báo lỗi.
Kết quả	Người dùng có thể xem lại thông tin về các chuyến đi đã thực hiện.

Bảng 9. Đặc tả chức năng “Quản lý thông tin”

Usecase	Quản lý thông tin
Tác nhân	Quản trị viên hoặc Người dùng (nếu tự quản lý)
Mô tả	Cho phép thực hiện các thao tác thêm, xem, sửa, xóa thông tin về một đối tượng cụ thể.
Điều kiện	1. Người dùng đã đăng nhập (nếu cần). 2. Người dùng có quyền thực hiện các thao tác quản lý thông tin.
Luồng chính	1. Người dùng chọn chức năng "Quản lý thông tin". 2. Hệ thống hiển thị danh sách các đối tượng thông tin hiện có. 3. Người dùng chọn một trong các thao tác: a. **Xem chi tiết:** Hệ thống hiển thị thông tin chi tiết của đối tượng được chọn. b. **Thêm mới:** i. Người dùng nhập thông tin mới. ii. Hệ thống kiểm tra tính hợp lệ của thông tin. iii. Người dùng xác nhận thêm mới. iv. Hệ thống lưu thông tin mới. c. **Chỉnh sửa:** i. Người dùng chọn đối tượng cần chỉnh sửa. ii. Hệ thống hiển thị thông tin hiện tại. iii. Người dùng chỉnh sửa thông tin. iv. Hệ thống kiểm tra tính hợp lệ của thông tin đã chỉnh sửa.

	v. Người dùng xác nhận chỉnh sửa. vi. Hệ thống cập nhật thông tin. d. **Xóa:** i. Người dùng chọn đối tượng cần xóa. ii. Hệ thống yêu cầu xác nhận xóa. iii. Người dùng xác nhận xóa. iv. Hệ thống xóa thông tin. 4. Hệ thống thông báo kết quả của thao tác (thành công/thất bại).
Luồng phụ	1. Người dùng chưa đăng nhập (nếu cần) -> Hệ thống yêu cầu đăng nhập. 2. Không tìm thấy đối tượng thông tin được chọn -> Hệ thống thông báo lỗi. 3. Thông tin nhập không hợp lệ -> Hệ thống thông báo lỗi và yêu cầu nhập lại. 4. Người dùng hủy thao tác -> Hệ thống quay lại màn hình trước đó. 5. Lỗi hệ thống (ví dụ: mất kết nối) -> Hệ thống thông báo lỗi. 6. (Tùy chọn) Một số điều kiện đặc biệt khi xóa (ví dụ: đối tượng đang được sử dụng ở nơi khác).
Kết quả	1. Thông tin được quản lý thành công (tùy theo thao tác cụ thể). 2. Hệ thống hiển thị thông báo xác nhận cho người dùng.

Bảng 10. Đặc tả chức năng “Gửi hồ sơ xác minh giấy tờ”

Usecase	Gửi hồ sơ xác minh giấy tờ
Tác nhân	Tài xế
Mô tả	Cho phép tài xế gửi các giấy tờ cần thiết để xác minh thông tin (ví dụ: ảnh giấy phép lái xe, đăng ký xe).
Điều kiện	Tài xế đã đăng nhập.
Luồng chính	1. Tài xế chọn chức năng "Gửi hồ sơ xác minh". 2. Hệ thống hiển thị giao diện cho phép tải lên các loại giấy tờ yêu cầu (ví dụ: giấy phép lái xe, đăng ký xe). 3. Tài xế tải lên các giấy tờ cần thiết.

	4. Tài xế xác nhận gửi hồ sơ. 5. Hệ thống nhận và lưu trữ hồ sơ, thông báo gửi thành công cho tài xế.
Luồng phụ	1. **Định dạng file không hợp lệ:** Hệ thống hiển thị thông báo lỗi và yêu cầu tải lại. 2. **Dung lượng file quá lớn:** Hệ thống hiển thị thông báo lỗi và yêu cầu tải lại. 3. **Lỗi kết nối:** Hệ thống hiển thị thông báo lỗi và hướng dẫn tài xế thử lại.
Kết quả	Hồ sơ được gửi thành công và được lưu trữ trong hệ thống để quản trị viên xem xét.

Bảng 11. Đặc tả chức năng “Cập nhật trạng thái chuyến đi”

Usecase	Cập nhật trạng thái chuyến đi
Tác nhân	Tài xế
Mô tả	Cho phép tài xế cập nhật thông tin về trạng thái thực tế của chuyến đi, bao gồm các giai đoạn như đến điểm đón, đón khách, trả khách, và các tình huống khác.
Điều kiện	1. Tài xế đang thực hiện một chuyến đi đã được phân công theo lịch trình. 2. Tài xế đã đăng nhập vào ứng dụng. 3. Hệ thống có chức năng theo dõi và cập nhật trạng thái chuyến đi.
Luồng chính	1. Tài xế chọn chuyến đi cần cập nhật trạng thái. 2. Hệ thống hiển thị giao diện cập nhật trạng thái, bao gồm các trạng thái có thể chọn. 3. Tài xế chọn trạng thái phù hợp với tình hình thực tế. 4. **Các trạng thái và hành động chi tiết:** a. **Đang đến điểm đón:** Tài xế chọn trạng thái "Đang đến điểm đón". b. **Đã đến điểm đón:** Tài xế chọn trạng thái "Đã đến điểm đón". c. **Đón khách:** Tài xế chọn trạng thái "Đón khách".

	d. **Đang di chuyển:** Trạng thái này có thể tự động cập nhật. e. **Đến điểm trả:** Tài xế chọn trạng thái "Đến điểm trả". f. **Trả khách:** Tài xế chọn trạng thái "Trả khách". g. **Hủy chuyến:** Tài xế chọn trạng thái "Hủy chuyến" và cung cấp lý do. h. **Trễ giờ:** Tài xế chọn trạng thái "Trễ giờ" và cung cấp thông tin chi tiết. 5. Hệ thống ghi nhận thời gian của mỗi lần cập nhật trạng thái. 6. (Tùy chọn) Hệ thống thông báo trạng thái mới cho khách hàng (nếu có) và quản trị viên.
Luồng phụ	1. **Lỗi kết nối:** Hệ thống hiển thị thông báo lỗi và yêu cầu tài xế kiểm tra kết nối. 2. **Không thể cập nhật trạng thái:** Hệ thống hiển thị thông báo lỗi và hướng dẫn tài xế chọn trạng thái phù hợp. 3. **Lỗi hệ thống chung:** Hệ thống hiển thị thông báo lỗi và hướng dẫn tài xế liên hệ hỗ trợ.
Kết quả	1. Hệ thống cập nhật thông tin về trạng thái chuyến đi. 2. Thông tin về thời gian và diễn biến của chuyến đi được ghi nhận. 3. Khách hàng và quản trị viên (nếu cần) được thông báo về trạng thái chuyến đi.

Bảng 12. Đặc tả chức năng “Xem chi tiết lộ trình chuyến đi”

Usecase	Xem chi tiết lộ trình chuyến đi
Tác nhân	Tài xế
Mô tả	Cho phép tài xế xem thông tin chi tiết về lộ trình của chuyến đi được phân công.
Điều kiện	1. Tài xế đã đăng nhập. 2. Tài xế đã được phân công chuyến đi.
Luồng chính	1. Tài xế chọn chuyến đi từ danh sách. 2. Hệ thống hiển thị màn hình "Chi tiết lộ trình". 3. Hệ thống hiển thị:

	a. Bản đồ tổng quan lộ trình. b. Danh sách chi tiết các điểm đón trả (địa chỉ, thời gian, thông tin hành khách). c. Các thông tin bổ sung (quãng đường, thời gian, v.v.). 4. Tài xế tương tác với bản đồ và xem chi tiết các điểm.
Luồng phụ	1. Không có thông tin hành khách -> Hiển thị điểm đón trả và số lượng. 2. Lỗi hiển thị bản đồ -> Hiển thị lộ trình dạng text. 3. (Nếu có) Thay đổi lộ trình -> Hệ thống cập nhật và hiển thị lộ trình mới.
Kết quả	Tài xế nắm rõ thông tin lộ trình và các điểm đón trả.

Bảng 13. Đặc tả chức năng “Nạp/rút tiền ví điện tử”

Usecase	Nạp/rút tiền ví điện tử
Tác nhân	Tài xế
Mô tả	Cho phép tài xế quản lý số dư và thực hiện giao dịch nạp/rút tiền trong ví điện tử.
Điều kiện	1. Tài xế đã đăng nhập. 2. Ví điện tử có thể có số dư tối thiểu. 3. Có thể có quy định về số tiền rút tối thiểu.
Luồng chính	**Nạp tiền:** 1. Tài xế chọn chức năng "Nạp tiền". 2. Hệ thống hiển thị các phương thức nạp tiền. 3. Tài xế chọn phương thức và thực hiện thanh toán. 4. Hệ thống cập nhật số dư ví điện tử. **Rút tiền:** 1. Tài xế chọn chức năng "Rút tiền". 2. Tài xế nhập thông tin rút tiền. 3. Tài xế xác nhận yêu cầu. 4. Hệ thống xử lý yêu cầu và cập nhật lịch sử giao dịch.

Luồng phụ	1. Thông tin nạp tiền không hợp lệ -> Yêu cầu nhập lại. 2. Thông tin rút tiền không hợp lệ -> Từ chối yêu cầu và thông báo lỗi.
Kết quả	Giao dịch nạp/rút tiền thành công, số dư ví điện tử được cập nhật.

Bảng 14. Đặc tả chức năng “Quản lý tài khoản người dùng”

Usecase	Quản lý tài khoản người dùng
Tác nhân	Quản trị viên
Mô tả	Cho phép Quản trị viên thực hiện các thao tác thêm, xem, sửa, xóa và chặn/mở khóa tài khoản của người dùng (khách hàng và tài xế).
Điều kiện	1. Quản trị viên đã đăng nhập. 2. Quản trị viên có quyền quản lý tài khoản người dùng. 3. Hệ thống có chức năng quản lý tài khoản người dùng.
Luồng chính	1. Quản trị viên chọn chức năng "Quản lý tài khoản người dùng". 2. Hệ thống hiển thị danh sách tài khoản người dùng. 3. Quản trị viên chọn một trong các thao tác: a. **Xem chi tiết:** Xem thông tin tài khoản. b. **Thêm mới:** Thêm tài khoản mới. c. **Chỉnh sửa:** Chỉnh sửa thông tin tài khoản. d. **Xóa:** Xóa tài khoản. e. **Chặn/Mở khóa:** Chặn hoặc mở khóa tài khoản.
Luồng phụ	1. Lỗi đăng nhập (Quản trị viên). 2. Không có quyền truy cập. 3. Lỗi nhập liệu. 4. Tài khoản không tồn tại. 5. Xung đột dữ liệu. 6. Lỗi hệ thống. 7. Xác nhận xóa.
Kết quả	Quản trị viên có thể quản lý tài khoản người dùng hiệu quả.

Bảng 15. Đặc tả chức năng “Duyệt hồ sơ tài xế”

Usecase	Duyệt hồ sơ tài xế
Tác nhân	Quản trị viên
Mô tả	Cho phép Quản trị viên có quyền duyệt tài khoản tài xế và kiểm tra hồ sơ của họ.
Điều kiện	Quản trị viên đã đăng nhập hệ thống quản trị.
Luồng chính	Quản trị viên chọn chức năng duyệt hồ sơ tài xế.,Hệ thống hiển thị danh sách các hồ sơ tài xế cần duyệt.,Quản trị viên xem chi tiết từng hồ sơ.,Quản trị viên quyết định duyệt hoặc từ chối hồ sơ.,Hệ thống cập nhật trạng thái hồ sơ và thông báo cho tài xế.
Luồng phụ	Không có hồ sơ nào cần duyệt -> Hệ thống hiển thị thông báo.,Hồ sơ không hợp lệ -> Hệ thống yêu cầu tài xế bổ sung thông tin.
Kết quả	Hồ sơ tài xế được duyệt hoặc từ chối thành công.

Bảng 16. Đặc tả chức năng “Quản lý tuyến đường”

Usecase	Quản lý tuyến đường
Tác nhân	Quản trị viên
Mô tả	Cho phép Quản trị viên thực hiện các thao tác thêm, sửa, xóa và quản lý thông tin về các tuyến đường, lịch trình và giá vé.
Điều kiện	1. Quản trị viên đã đăng nhập vào hệ thống. 2. Hệ thống có chức năng quản lý tuyến đường.
Luồng chính	1. Quản trị viên chọn chức năng "Quản lý tuyến đường". 2. Hệ thống hiển thị danh sách các tuyến đường hiện có. 3. Quản trị viên chọn một trong các thao tác: a. **Thêm mới:** i. Quản trị viên nhập thông tin tuyến đường mới (điểm đầu, điểm cuối, các điểm dừng, lịch trình, giá vé). ii. Hệ thống kiểm tra tính hợp lệ của thông tin. iii.Hệ thống lưu thông tin tuyến đường mới. b. **Chỉnh sửa:** i. Quản trị viên chọn tuyến đường cần chỉnh sửa. ii. Hệ thống hiển thị thông tin tuyến đường hiện tại.

	iii.Quản trị viên chỉnh sửa thông tin. iv.Hệ thống kiểm tra tính hợp lệ của thông tin đã chỉnh sửa. v. Hệ thống cập nhật thông tin tuyến đường. c. **Xóa:** i. Quản trị viên chọn tuyến đường cần xóa. ii. Hệ thống hiển thị yêu cầu xác nhận xóa. iii.Quản trị viên xác nhận xóa. iv.Hệ thống xóa thông tin tuyến đường.
Luồng phụ	1. Thông tin nhập vào không hợp lệ -> Hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại. 2. Không tìm thấy tuyến đường cần chỉnh sửa/xóa -> Hệ thống hiển thị thông báo lỗi. 3. Lỗi hệ thống trong quá trình thêm/sửa/xóa -> Hệ thống hiển thị thông báo lỗi.
Kết quả	Quản trị viên có thể quản lý thông tin tuyến đường một cách hiệu quả.

Bảng 17. Đặc tả chức năng “Phân công tài xế”

Usecase	Phân công tài xế
Tác nhân	Quản trị viên
Mô tả	Cho phép Quản trị viên thực hiện việc phân công tài xế cho các tuyến đường hoặc các chuyến xe cụ thể.
Điều kiện	1. Quản trị viên đã đăng nhập. 2. Hệ thống có thông tin về tuyến đường/chuyến xe và danh sách tài xế. 3. Các tài xế ở trạng thái sẵn sàng.
Luồng chính	1. Quản trị viên chọn chức năng "Phân công tài xế". 2. Hệ thống hiển thị giao diện phân công. 3. Quản trị viên chọn tuyến đường/chuyến xe. 4. Hệ thống hiển thị thông tin chi tiết về tuyến đường/chuyến xe. 5. Quản trị viên chọn tài xế.

	6. Hệ thống (tùy chọn) hiển thị gợi ý/cảnh báo. 7. Quản trị viên xác nhận phân công. 8. Hệ thống cập nhật thông tin và thông báo thành công. 9. Hệ thống (tùy chọn) gửi thông báo cho tài xế.
Luồng phụ	1. Không có tuyến đường/chuyến xe để phân công. 2. Không có tài xế nào sẵn sàng. 3. Tài xế không phù hợp. 4. Xung đột lịch trình. 5. Lỗi hệ thống.
Kết quả	Tài xế được phân công thành công, thông tin phân công được lưu trữ.

Bảng 18. Đặc tả chức năng "Cấu hình hệ thống"

Usecase	Cấu hình hệ thống
Tác nhân	Quản trị viên
Mô tả	Cho phép Quản trị viên xem hoặc thực hiện các cấu hình liên quan đến phí, giá cước, thuế, các giới hạn của hệ thống.
Điều kiện	Quản trị viên đã đăng nhập
Luồng chính	Quản trị viên chọn chức năng cấu hình hệ thống.,Hệ thống hiển thị các tùy chọn cấu hình (phí, giá cước, thuế, giới hạn).,Quản trị viên thay đổi các thông số cấu hình.,Hệ thống lưu lại các thay đổi.
Luồng phụ	Thông tin cấu hình không hợp lệ -> Hệ thống hiển thị thông báo lỗi.
Kết quả	Hệ thống được cấu hình theo yêu cầu.

Bảng 19. Đặc tả chức năng "Xem thống kê"

Usecase	Xem thống kê
Tác nhân	Quản trị viên
Mô tả	Cho phép Quản trị viên xem các số liệu thống kê về hoạt động của hệ thống (ví dụ: số lượng đặt xe, doanh thu, v.v.)
Điều kiện	1. Quản trị viên đã đăng nhập. 2. Hệ thống có chức năng thống kê và lưu trữ dữ liệu cần thiết.

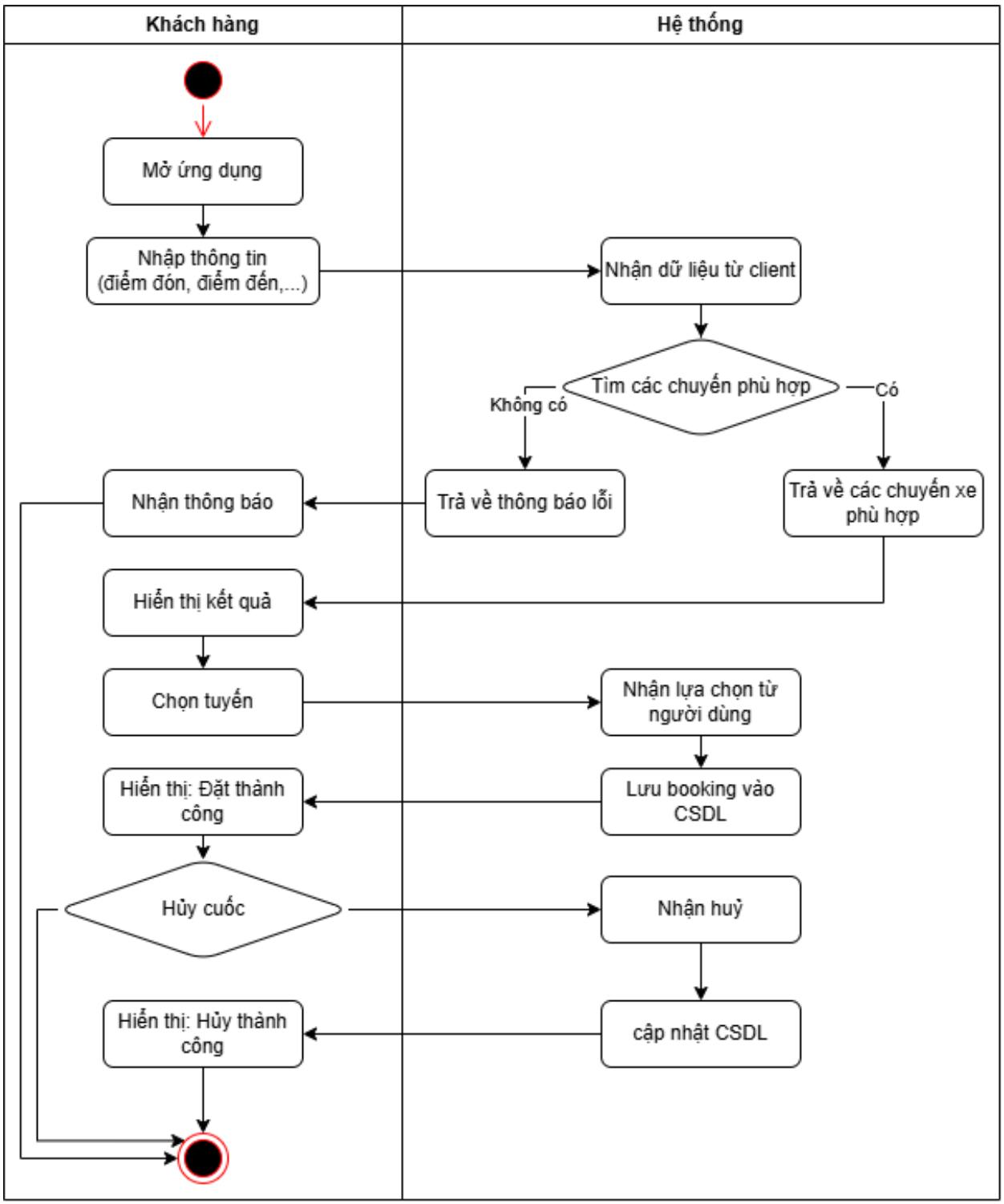
Luồng chính	1. Quản trị viên chọn chức năng "Xem thống kê". 2. Hệ thống hiển thị các loại thống kê có thể xem (ví dụ: Thống kê theo ngày, Thống kê theo tháng, Thống kê theo tài xế, Thống kê theo tuyến đường). 3. Quản trị viên chọn loại thống kê mong muốn. 4. Hệ thống truy xuất dữ liệu thống kê từ cơ sở dữ liệu. 5. Hệ thống hiển thị thông tin thống kê (có thể dưới dạng bảng, biểu đồ). 6. (Tùy chọn) Quản trị viên có thể sử dụng các chức năng lọc, chọn khoảng thời gian.
Luồng phụ	1. Quản trị viên chưa đăng nhập -> Hệ thống yêu cầu đăng nhập. 2. Lỗi khi truy xuất dữ liệu (ví dụ: mất kết nối, lỗi truy vấn) -> Hệ thống thông báo lỗi. 3. Không có dữ liệu cho loại thống kê đã chọn -> Hệ thống thông báo không có dữ liệu
Kết quả	1. Hệ thống hiển thị thông tin thống kê dưới dạng phù hợp (bảng, biểu đồ). 2. Quản trị viên có thể xem và phân tích các số liệu thống kê về hoạt động của hệ thống.

Bảng 20. Đặc tả chức năng “Trợ giúp và phản hồi”

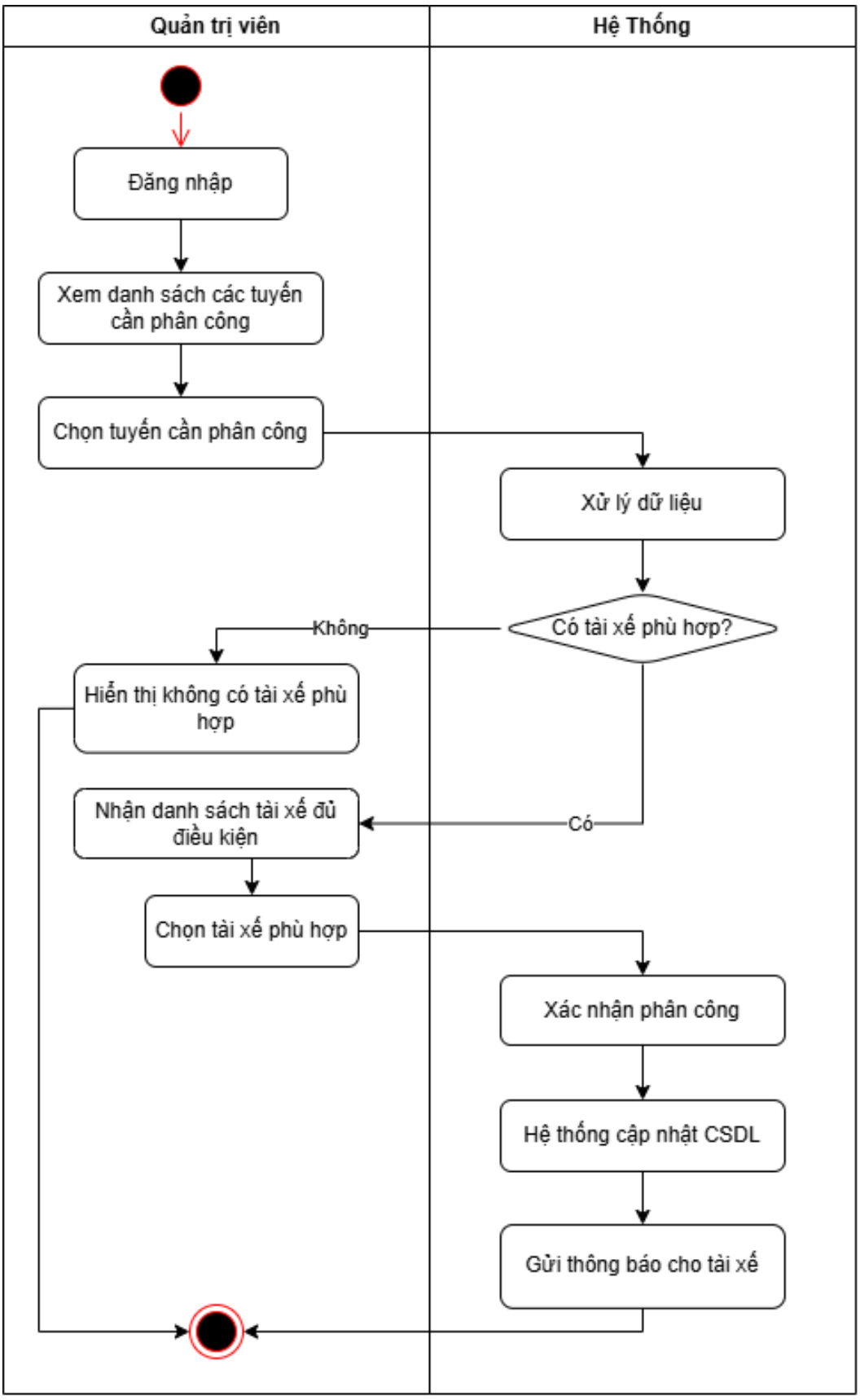
Usecase	Trợ giúp và phản hồi
Tác nhân	Quản trị viên
Mô tả	Quản trị viên xem báo cáo của người dùng và xử lý vấn đề
Điều kiện	Quản trị viên đã đăng nhập
Luồng chính	1. Xem danh sách báo cáo 2. Xem chi tiết báo cáo 3. Phản hồi báo cáo
Luồng phụ	Không có
Kết quả	Phản hồi hỗ trợ thành công

2.4. Biểu đồ hoạt động

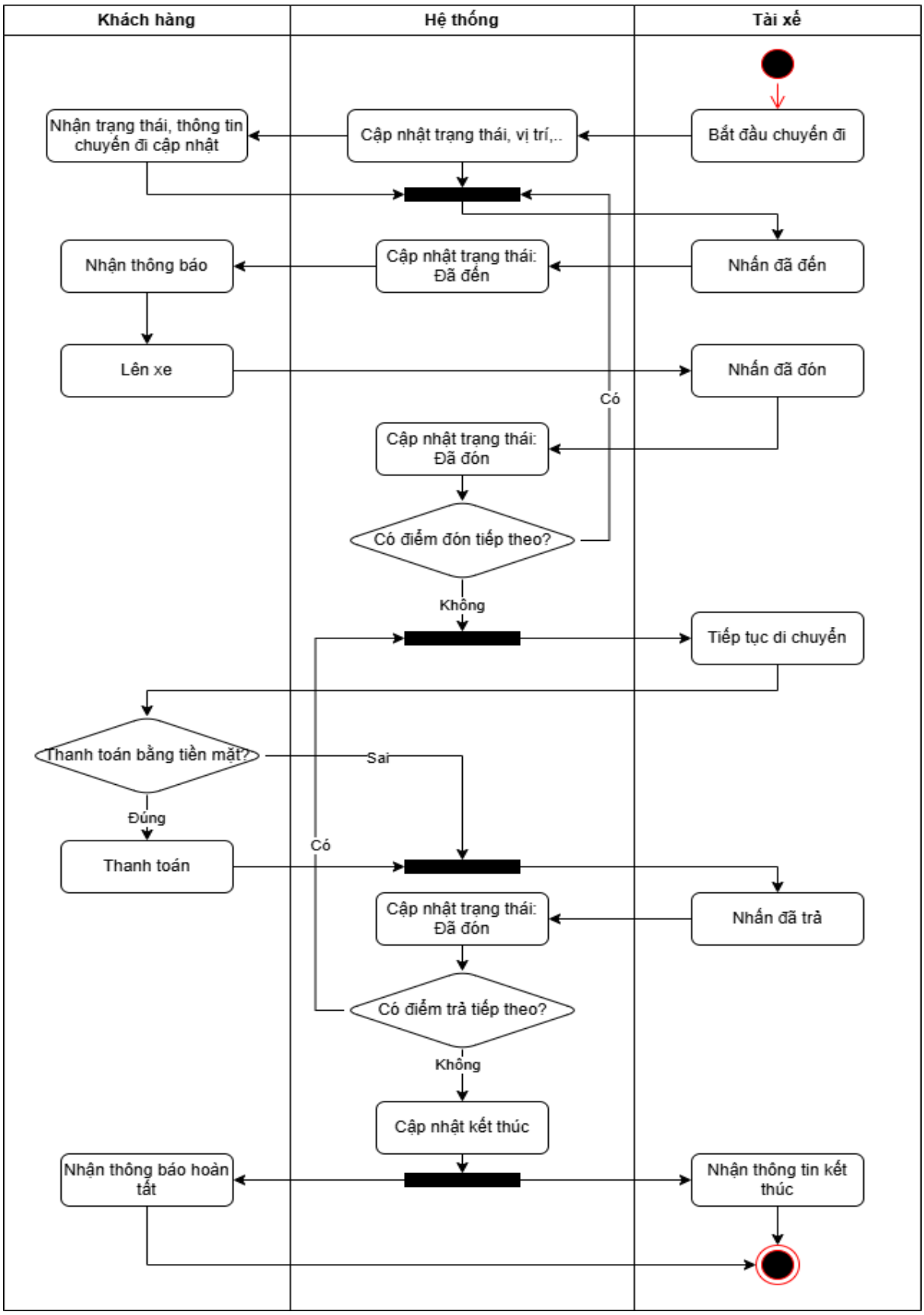
Để minh họa chi tiết luồng nghiệp vụ của hệ thống vận tải hành khách ghép tuyến, đặc biệt là sự tương tác giữa các tác nhân với nhau và với Hệ thống, phần này sẽ trình bày các biểu đồ hoạt động (Activity Diagram). Dưới đây là các biểu đồ hoạt động mô tả các quy trình cốt lõi của hệ thống vận tải hành khách ghép tuyến, bao gồm quy trình đặt chuyến tổng quát, quy trình phân công chuyến đi, và quy trình kết thúc chuyến đi.



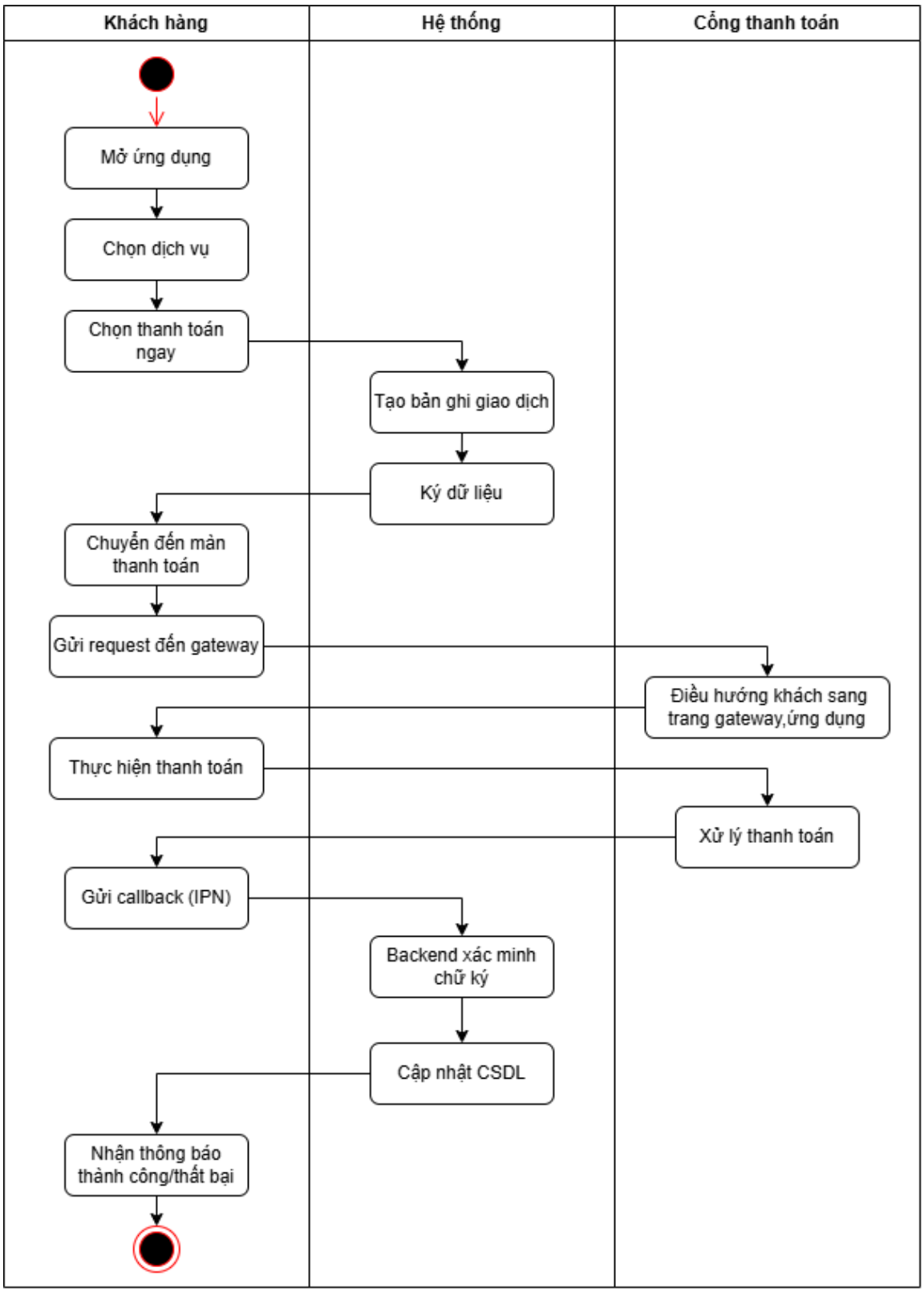
Hình 6. Biểu đồ luồng hoạt động nghiệp vụ "Đặt cuốc xe"



Hình 7. Biểu đồ luồng hoạt động nghiệp vụ "Phân công tài xế"



Hình 8. Biểu đồ luồng hoạt động nghiệp vụ "Đón trả khách"



Hình 9. Biểu đồ luồng hoạt động nghiệp vụ "Thanh toán online"

2.5. Thiết kế cơ sở dữ liệu

Phần này trình bày cấu trúc cơ sở dữ liệu của hệ thống, tập trung vào tổ chức dữ liệu trên MongoDB – hệ quản trị NoSQL chính. Thiết kế ưu tiên tính linh hoạt, hiệu suất và

khả năng mở rộng, phù hợp với dữ liệu động của ứng dụng vận tải hành khách ghép tuyến. Dữ liệu được lưu trữ dưới dạng tài liệu (document) trong các collection, sử dụng tham chiếu (referencing) hoặc nhúng (embedding) tùy theo yêu cầu nghiệp vụ.

Hệ thống tổ chức dữ liệu thành các collection chính, phản ánh các thực thể cốt lõi và mối quan hệ giữa chúng. Dưới đây là cấu trúc các collection chính, bao gồm các trường quan trọng và mục đích sử dụng:

Bảng 21. Thiết kế cơ sở dữ liệu

STT	**Collection**	**Mô tả và Cấu trúc**

1	Account	// Lưu thông tin tài khoản người dùng.
		{
		_id: ObjectId,
		name: String,
		phone: String,
		avatar: String
		password: String,
		role: String (“rider” | “driver” | “admin” | “owner”),
		status: String ("unverified" | "pending" | "active" | "banned"),
		createdAt: Date,
		updatedAt: Date
		}

2	Driver	// Lưu hồ sơ tài xế, tham chiếu đến Account.
		{
		_id: ObjectId,
		accountId: ObjectId (ref Account),
		number: Number,
		documents: [
		{
		name: String,
		document: [String],
		status: String ("pending" | "verified" | "exprired"),
		note: String,

		expire: Date, },], }

3	Vehicle	// Thông tin phương tiện của tài xế.
		{
		_id: ObjectId,
		driverId: ObjectId (ref Driver),
		type: String,
		make: String,
		model: String,
		color: String,
		seat: Number,
		licensePlate: String,
		vehicleImage: String (URL từ Firebase Storage)
		}

4	Route	// Tuyến cố định do admin thiết lập.
		{
		_id: ObjectId,
		name: String,
		from: GeoJson,
		to: GeoJson,
		isActive: Boolean
		}

5	Assignment	// Lịch phân công tài xế cho tuyến.
		{
		_id: ObjectId,
		routeId: ObjectId (ref Router),
		driverId: ObjectId (ref Driver),
		schedule: {
		type: String ("even" | "odd" | "daily" | "custom"),

		time: String, days: [Number] }, isActive: Boolean }

6	Trip	// Thông tin chuyến đi cụ thể.
		{
		_id: ObjectId,
		routeId: ObjectId (ref Route),
		driverId: ObjectId (ref Driver),
		departureTime: Date,
		arriveTime:Date
		startLocation: GeoJson;
		endLocation: GeoJson;
		waypoints: GeoJson[];
		bookings: ObjectId[] (ref Booking)
		distance: number;
		duration: number;
		history: string; (Ghi chú các vấn đề của chuyến đi)
		status: String ("scheduled" | "started" | "finished")
		}

7	Booking	// Đặt chỗ của hành khách trên chuyến đi.
		{
		_id: ObjectId,
		accountId: ObjectId (ref Account),
		pickup: GeoJson,
		dropoff: GeoJson,
		departure: Date,
		distance: Number,
		duration: Number,
		passengers: Number,

		priceInfo: String, price: Number, status: String ("pending" | "canceled" | "process" | "ending" | "finished"), paymentStatus:String ("pending" | " completed " | " failed "), paymentMethod: String (“cash”, ...); note: String; createdAt: Date }

8	Transaction	// Lịch sử giao dịch nạp/rút tiền.
		{
		_id: ObjectId,
		accountId: ObjectId (ref Account),
		bookingId:ObjectId (ref Booking),
		amount: Number,
		type: String ("deposit" | "withdraw"),
		kind: String,
		method: String (“cash” | ...),
		status: String ("pending" | "completed" | "failed"),
		createdAt: Date
		}

9	Notification	// Lưu thông báo hệ thống.
		{
		_id: ObjectId,
		accountId: ObjectId (ref Account),
		type: String,
		title: string;
		content: string;
		image: string;
		isRead: Boolean,
		createdAt: Date,

		expireAt: Date }

10	Rating	// Đánh giá tài xế từ khách hàng sau chuyến đi.
		{
		_id: ObjectId,
		tripId: ObjectId (ref Trip),
		userId: ObjectId (ref Account),
		driverId: ObjectId (ref Driver),
		rating: Number (1-5),
		comment: String (optional),
		createdAt: Date
		}

11	Config	// Cấu hình hệ thống (giá cước, thuế, phí).
		{
		_id: ObjectId,
		type: String,
		value: Mixed (String, Number, Object),
		description: String,
		condition: String,
		updatedAt: Date
		}

12	Help	// Quản lý yêu cầu hỗ trợ và phản hồi.
		{
		_id: ObjectId,
		userId: ObjectId (ref Account),
		issue: String, description: String,
		status: String ("open" | "in_progress" | "resolved"),
		response: String (optional),
		createdAt: Date,
		updatedAt: Date
		}

Bảng 22. Một số loại dữ liệu chính khác.

Location	// Vị trí thời gian thực của tài xế sẽ được lưu trữ ở redis
	{
	_id: string,
	accountId: ObjectId (ref Account),
	location: GeoJson,
	timestamp: Date,
	}

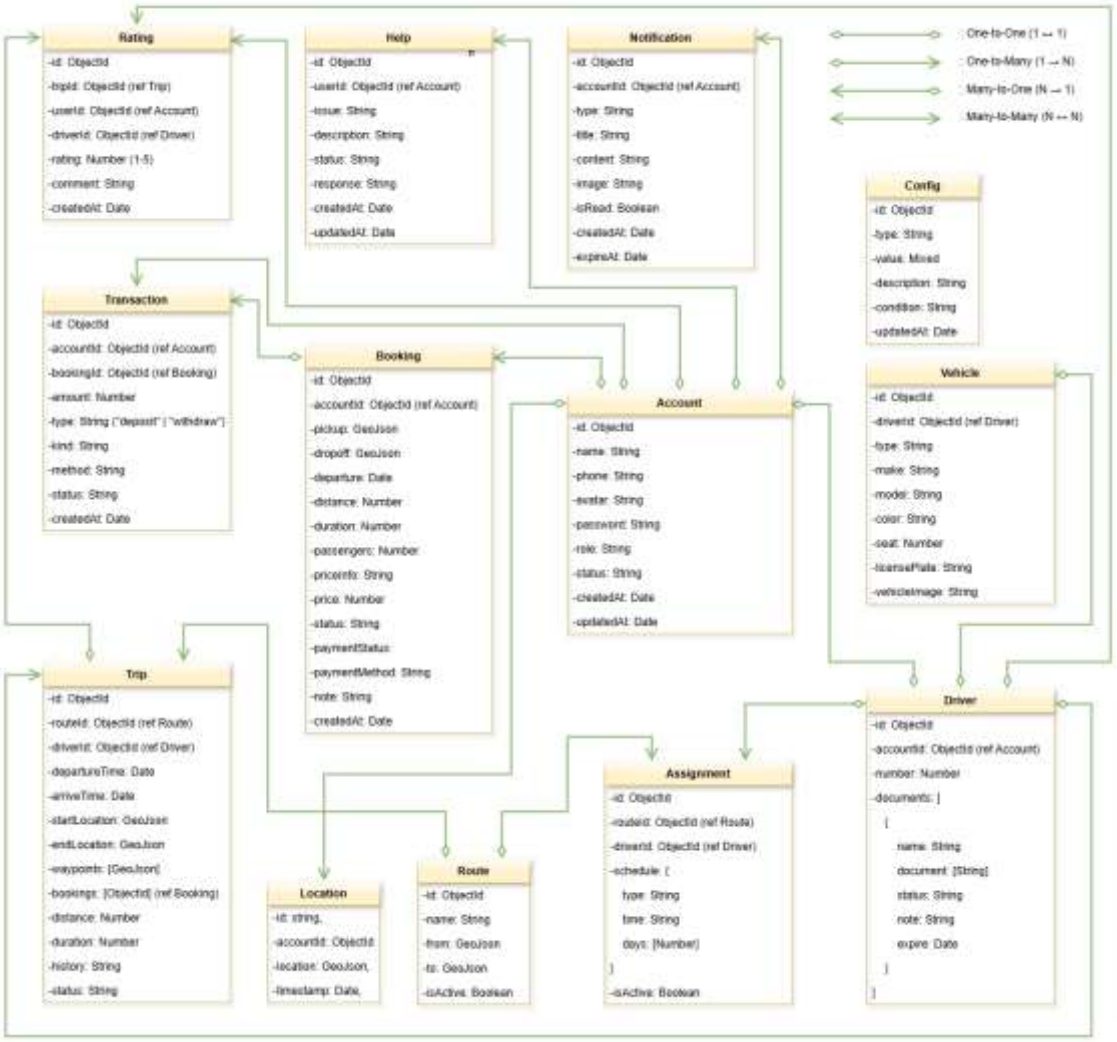
GeoJson	// Tọa độ và các thông tin khác
	{
	type: string ("Feature" | ...),
	geometry: {
	type: string ("Point" | ...),
	coordinates:
	| [number, number] // Point: [longitude, latitude]
	| [number, number][]
	| ...
	};
	properties: {
	name?: string; // Tên địa điểm
	address?: string; // Địa chỉ
	city?: string; // Thành phố
	country?: string; // Quốc gia
	postalCode?: string; // Mã bưu điện
	description?: string; // Mô tả thêm
	icon?: string; // URL hoặc tên icon cho bản đồ
	id?: string | number; // ID duy nhất cho feature,
	chủ yếu lưu place_id từ goong map
	[key: string]: any; // Cho phép mở rộng thuộc tính tùy chỉnh
	};
	}

Mối quan hệ giữa các collection:

- Một Account có thể liên kết với một Driver (quan hệ 1-1, nếu người dùng là tài xế). Quan hệ này được thể hiện qua trường AccountId trong Driver tham chiếu đến _id của Account.
- Một Driver được gán nhiều Assignment (1-nhiều) và thực hiện nhiều Trip (1-nhiều). Quan hệ này sử dụng tham chiếu driverId trong Assignment và Trip.
- Một Trip có nhiều Booking (1-nhiều) và liên quan đến một Transaction (1-1). Quan hệ Trip-Booking được thể hiện qua tripId trong Booking.
- Một Account nhận nhiều Notification (1-nhiều), thể hiện qua AccountId trong Notification.
- Một Driver liên kết với nhiều Location (1-nhiều) để theo dõi vị trí thời gian thực, thông qua AccountId trong Location.

Nguyên tắc Thiết kế dữ liệu:

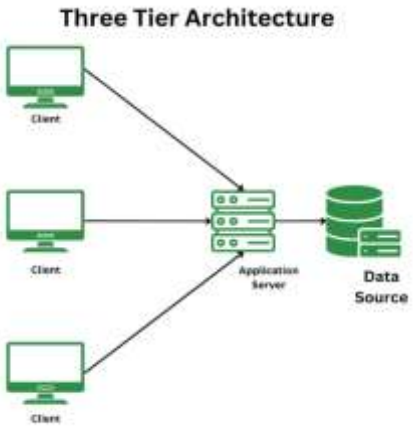
- **Tách biệt thực thể**: Phân tách Account và Driver để quản lý thông tin rõ ràng; Route và Assignment tách biệt để linh hoạt phân công.
- **Tham chiếu và nhúng**: Sử dụng tham chiếu (ref) cho mối quan hệ 1-nhiều (user-trip, trip-booking); nhúng dữ liệu phụ (waypoints trong Trip) để tăng hiệu suất truy vấn.
- **Linh hoạt**: Thiết kế schema hỗ trợ thêm trường mới mà không cần thay đổi cấu trúc hiện có.
- **Tích hợp với công nghệ khác**: **Redis**: Lưu trữ dữ liệu tạm (vị trí tài xế, trạng thái chuyến đi) từ các collection như trips, drivers. **Firebase Storage**: Lưu trữ tệp (ảnh giấy tờ, phương tiện), tham chiếu qua URL trong drivers, vehicles.



Hình 10. Sơ đồ ERD

2.6. Kiến trúc hệ thống

Kiến trúc hệ thống đóng vai trò nền tảng, định hình cách các thành phần tương tác để hệ thống vận tải hành khách ghép tuyến hoạt động hiệu quả trên nhiều nền tảng. Theo **kiến trúc** Client–Server kết hợp mô hình ba lớp (3-tier Architecture), hệ thống được phân tách rõ ràng thành các lớp trình bày, logic nghiệp vụ và dữ liệu, được thể hiện trong hình sau



Hình 11. Sơ đồ Kiến trúc Hệ thống

2.6.1. Lớp trình bày (Presentation Layer)

Lớp trình bày đóng vai trò giao tiếp trực tiếp với người dùng, cung cấp giao diện trực quan để thu thập dữ liệu đầu vào và hiển thị kết quả xử lý từ hệ thống. Lớp này được thiết kế tách biệt hoàn toàn với lớp logic nghiệp vụ, đảm bảo tính mô-đun, dễ bảo trì và mở rộng. Hệ thống hỗ trợ ba giao diện chính, phục vụ ba nhóm người dùng: hành khách, tài xế và quản trị viên.

a, Ứng dụng Di động (Rider App và Driver App):

- **Mục đích**: Cung cấp giao diện thân thiện cho hành khách (đặt chuyến, theo dõi hành trình) và tài xế (nhận chuyến, quản lý lộ trình).
- **Công nghệ**: React Native, sử dụng Expo để tối ưu phát triển đa nền tảng (iOS và Android).
- **Kiến trúc chính**:
  - **Giao diện người dùng (UI)**: Các thành phần UI gốc (native components) đảm bảo hiệu năng cao, tích hợp Google Maps SDK cho bản đồ và định vị thời gian thực.
  - **Tương tác dữ liệu**: Giao tiếp với backend qua API REST (HTTP/HTTPS) bằng thư viện Axios. Socket.io được sử dụng để cập nhật trạng thái chuyến đi và vị trí xe theo thời gian thực.
  - **Tính năng chính**: Đăng nhập/đăng ký, đặt/nhận chuyến, theo dõi lộ trình, thanh toán, quản lý hồ sơ, lịch sử giao dịch.
- **Ưu điểm**: Đơn codebase, hỗ trợ hot-reloading, dễ tích hợp thư viện bên thứ ba (react-native-maps,...).

b, Ứng dụng Web Quản trị (Admin Panel):

- **Mục đích**: Cung cấp công cụ quản lý cho quản trị viên, bao gồm quản lý người dùng, chuyến đi, cấu hình hệ thống và báo cáo.
- **Công nghệ**: ReactJS kết hợp TypeScript, tối ưu giao diện động và bảo trì code.
- **Kiến trúc chính**:
  - **Giao diện người dùng (UI)**: Xây dựng theo mô hình component-based, hỗ trợ dashboard tổng quan, bảng biểu và biểu đồ phân tích dữ liệu.
  - **Tương tác dữ liệu**: Kết nối với backend qua API REST (HTTP/HTTPS), đảm bảo phản hồi nhanh và xử lý bất đồng bộ.
  - **Tính năng chính**: Quản lý tài khoản, chuyến đi, cấu hình giá cước, báo cáo thống kê.
- **Ưu điểm**: Tái sử dụng code, giao diện trực quan, dễ mở rộng và tích hợp công cụ phân tích (chart libraries).

c, Đặc điểm kiến trúc

- **Tách biệt tầng**: Lớp trình bày độc lập với logic nghiệp vụ, cho phép thay đổi giao diện mà không ảnh hưởng đến backend.
- **Tính mở rộng**: Sử dụng framework hiện đại (React Native, ReactJS) với cộng đồng hỗ trợ lớn, dễ tích hợp tính năng mới.
- **Hiệu năng**: Tối ưu hóa trải nghiệm người dùng qua xử lý bất đồng bộ, kết nối thời gian thực và giao diện responsive.
- **Bảo mật**: Tất cả giao tiếp với backend sử dụng HTTPS, tích hợp xác thực token-based (JWT) cho đăng nhập/đăng ký.

2.6.2. Lớp xử lý nghiệp vụ (Application Layer)

Lớp Logic Nghiệp vụ, hay Lớp Ứng dụng, đóng vai trò trung tâm trong hệ thống, xử lý các yêu cầu từ Lớp Trình bày, thực thi logic nghiệp vụ phức tạp và tương tác với Lớp Dữ liệu. Lớp này được thiết kế theo mô hình MVC, tách biệt hoàn toàn với giao diện người dùng và cơ sở dữ liệu, đảm bảo tính mô-đun, dễ mở rộng và bảo trì. Thay vì trả về giao diện HTML, lớp này tập trung vào xử lý dữ liệu và trả về JSON cho các ứng dụng client.

a, Nền tảng và Công nghệ chính

- **Node.js với Express.js**: Sử dụng Node.js để xử lý bất đồng bộ, phù hợp cho ứng dụng thời gian thực và tải cao. Express.js cấu trúc API RESTful, quản lý tuyến đường (routes), middleware và controllers.
- **Socket.io**: Tích hợp để hỗ trợ giao tiếp hai chiều (bidirectional), phục vụ cập nhật vị trí xe, trạng thái chuyến đi và thông báo thời gian thực.
- **Mongoose (ODM for MongoDB)**: Định nghĩa schema dữ liệu và thực hiện các thao tác CRUD trên MongoDB, đảm bảo tính nhất quán và hiệu quả.

b, Mô hình MVC

- **Controller**: Điều phối yêu cầu từ Lớp Trình bày, không xử lý trực tiếp dữ liệu mà gọi Model để thực thi logic nghiệp vụ, sau đó trả về JSON response.
- **Model**: Quản lý dữ liệu và logic nghiệp vụ cơ bản, tương tác với MongoDB thông qua Mongoose.
- **View**: Thay thế bằng JSON response thay vì giao diện HTML, phù hợp với kết nối RESTful với frontend.

c, Các Thành phần và Module chính

Lớp Logic Nghiệp vụ được tổ chức thành các module chuyên biệt:

- **API Gateway/Routing**:
  - Tiếp nhận yêu cầu HTTP/HTTPS từ Lớp Trình bày.
  - Điều hướng đến các controller tương ứng, đảm bảo tổ chức endpoint rõ ràng và bảo mật.
- **Authentication & Authorization**:
  - Xác thực đăng nhập/đăng ký (hành khách, tài xế, admin) bằng JWT.
  - Kiểm soát quyền truy cập dựa trên vai trò (role-based access control).
- **User/Driver Management:**
  - Quản lý hồ sơ (tạo, cập nhật, xóa, truy vấn) và trạng thái của người dùng/tài xế.
  - Quản lý tuyến xe và phân công tài xế.
- **Ride Management**:
  - Xử lý yêu cầu đặt chuyến, tìm tài xế phù hợp bằng thuật toán tối ưu (dựa trên khoảng cách).
  - Quản lý trạng thái chuyến đi (đang tìm, đã chấp nhận, đang di chuyển, hoàn thành, hủy).

- Tính toán giá cước dựa trên khoảng cách, thời gian và chính sách hệ thống.

- **Real-time Location & Notification**:
  - Cập nhật vị trí tài xế theo thời gian thực qua Socket.io.
  - Gửi thông báo đẩy (FCM) và thông báo tức thời cho hành khách/tài xế.
- **Payment Processing**:
  - Tích hợp cổng thanh toán (Vnpay, MoMo) để xử lý giao dịch.
  - Ghi nhận và đối soát thanh toán.
- **Logging & Error Handling**:
  - Ghi log sự kiện và lỗi để giám sát, gỡ lỗi.
  - Xử lý ngoại lệ, trả về phản hồi lỗi chuẩn cho client.

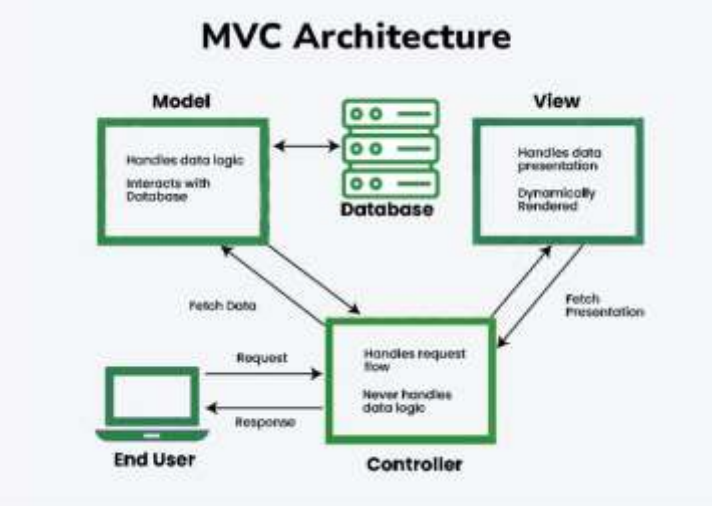
d, Tương tác với Dịch vụ bên ngoài

- **Goong Maps APIs**: Hỗ trợ geocoding, tính toán khoảng cách/thời gian, và định tuyến.
- **Payment Gateway APIs**: Kết nối với MoMo, VnPay cho giao dịch trực tuyến.
- **SMS Gateway**: Gửi tin nhắn xác nhận hoặc thông báo quan trọng.
- **FCM (Firebase Cloud Messaging)**: Đẩy thông báo đến ứng dụng di động.

e, Đặc điểm Kiến trúc

- **Tách biệt tầng**: Controller độc lập với Model và View (JSON response), đảm bảo dễ bảo trì và mở rộng.
- **Hiệu suất**: Tận dụng bất đồng bộ của Node.js và Socket.io cho xử lý thời gian thực.
- **Mở rộng**: Sử dụng MongoDB với Mongoose, hỗ trợ linh hoạt với dữ liệu không cố định (schema-less).
- **Bảo mật**: Áp dụng mã hóa JWT, kiểm tra quyền truy cập và log lỗi chi tiết.

Lớp Logic Nghiệp vụ là cầu nối giữa Lớp Trình bày và Lớp Dữ liệu, đảm bảo hệ thống xử lý hiệu quả các yêu cầu nghiệp vụ phức tạp, duy trì tính toàn vẹn dữ liệu và cung cấp trải nghiệm mượt mà cho người dùng.



Hình 12. Mô hình MVC

2.6.3. Lớp dữ liệu (Data Layer)

Lớp Dữ liệu là tầng thấp nhất trong kiến trúc MVC, chịu trách nhiệm lưu trữ, truy xuất và quản lý dữ liệu, cung cấp nguồn dữ liệu ổn định, toàn vẹn và an toàn cho Lớp Logic Nghiệp vụ (Controller). Lớp này được thiết kế tách biệt hoàn toàn với logic ứng dụng, đảm bảo tính mô-đun và khả năng thay thế công nghệ.

a, Thành phần và Công nghệ Chính

- **MongoDB**:
  - Hệ quản trị NoSQL, lưu trữ dữ liệu dạng document (BSON).
  - Triển khai trong Docker với chế độ replica set, đảm bảo tính sẵn sàng và mở rộng ngang.
  - Tích hợp qua Mongoose (ODM) để Controller thực hiện các thao tác CRUD.
- **Redis**:
  - Hệ thống cache và lưu trữ tạm, hỗ trợ truy xuất nhanh cho dữ liệu thời gian thực (vị trí tài xế, token, trạng thái chuyến đi).
  - Giảm tải cho MongoDB, tăng hiệu suất xử lý.
- **Firebase Storage & Authentication**:
  - **Firebase Storage**: Lưu trữ tệp nhị phân (ảnh tài xế, giấy tờ xác minh).
  - **Firebase Cloud Messaging (FCM)**: Gửi thông báo đẩy tới client.
  - **Firebase Authentication**: Hỗ trợ xác thực OTP và số điện thoại.
- **Goong API/Google Maps API**:
  - Cung cấp dịch vụ bản đồ cho tính toán khoảng cách, định vị GPS và gợi ý tuyến đường.

b, Vai trò trong MVC

- **Model**: Đóng vai trò Model, ánh xạ dữ liệu từ các nguồn (MongoDB, Redis) thành đối tượng JavaScript và ngược lại.
- **Tương tác**: Nhận yêu cầu từ Controller, thực thi truy vấn hoặc lưu trữ, trả kết quả mà không giao tiếp trực tiếp với View.

c, Cơ chế Tương tác

- **MongoDB**: Controller sử dụng Mongoose để truy vấn dữ liệu, tận dụng replica set cho tính sẵn sàng.
- **Redis**: Lưu trữ dữ liệu tạm với cấu trúc key-value, hỗ trợ pub/sub cho thông báo thời gian thực.
- **Firebase Storage**: Cung cấp URL an toàn để Controller quản lý tệp (tải lên/tải xuống).
- **Goong/Google Maps API**: Controller gọi API để lấy dữ liệu bản đồ (khoảng cách, tuyến đường).

d, Đặc điểm Kiến trúc

- **Hiệu suất**: Redis tối ưu truy xuất thời gian thực; MongoDB replica set đảm bảo độ tin cậy.
- **Mở rộng**: MongoDB hỗ trợ sharding; Firebase Storage tự động mở rộng.
- **Bảo mật**: MongoDB và Firebase sử dụng mã hóa tại tầng lưu trữ; Redis áp dụng ACL.

Lớp Dữ liệu đảm bảo hệ thống vận hành ổn định, hiệu quả, đáp ứng nhu cầu lưu trữ và truy xuất dữ liệu phức tạp.

2.7. Thuật toán gợi ý và ghép tuyến

2.7.1. Mục tiêu

Thuật toán này được thiết kế để nâng cao hiệu quả vận hành và trải nghiệm người dùng trong hệ thống đặt xe chung. Các mục tiêu chính bao gồm:

- Gợi ý các chuyến xe phù hợp: Dựa trên các tiêu chí về vị trí đón/trả, thời gian khởi hành và số ghế trống, nhằm tối ưu hóa việc tìm kiếm cho người dùng.

- Ghép yêu cầu đặt chỗ (booking) vào chuyến xe: Tối ưu hóa lộ trình và quãng đường di chuyển của chuyến xe sau khi nhận thêm booking mới, đảm bảo hiệu suất hoạt động.
- Đảm bảo tính chính xác và nhất quán dữ liệu: Cung cấp thông tin quãng đường, thời gian, giá tiền một cách chính xác và duy trì tính toàn vẹn của dữ liệu trong toàn bộ quá trình.

2.7.2. Quy trình thuật toán

Quy trình thuật toán được chia thành hai phần chính: "Thuật toán gợi ý tuyến" và "Thuật toán ghép tuyến".

A, Thuật toán gợi ý tuyến

Thuật toán này có nhiệm vụ tìm kiếm và sắp xếp các chuyến xe tiềm năng cho một yêu cầu đặt chỗ cụ thể.

Input: Yêu cầu đặt chỗ bao gồm điểm đón/trả (tọa độ GeoJSON Point), thời gian dự kiến và số hành khách.

Các bước thực hiện:

1. Lọc chuyến xe sơ bộ: Truy vấn các chuyến xe (Trip) có ngày khởi hành trùng hoặc trong khoảng ±30 phút so với thời gian yêu cầu. Chỉ xem xét các chuyến có trạng thái "sẵn sàng nhận khách" và còn đủ ghế trống (số ghế trống ≥ số hành khách).
2. Kiểm tra khoảng cách gần đúng: Tính khoảng cách địa lý từ điểm đón của khách đến điểm bắt đầu tuyến và từ điểm trả của khách đến điểm kết thúc tuyến. Các chuyến xe phải có cả hai khoảng cách này dưới ngưỡng tối đa 2 km (đây là ngưỡng có thể cấu hình được trong hệ thống).
3. Gom nhóm tuyến (Clustering): Áp dụng thuật toán K-Means Clustering để nhóm các chuyến xe có điểm đi và điểm đến tương tự về mặt địa lý. Việc này giúp ưu tiên gợi ý các tuyến trong cùng khu vực hoặc hành trình.
4. Sắp xếp ưu tiên: Sắp xếp các chuyến xe đã được lọc và gom nhóm theo thứ tự ưu tiên (với trọng số giảm dần):
   - Khoảng cách điểm đón/trả ngắn nhất: Ưu tiên các chuyến xe có điểm đón/trả gần nhất với yêu cầu của khách.
   - Thời gian khởi hành gần nhất: Ưu tiên các chuyến xe có thời gian khởi hành gần với yêu cầu nhất.

- Giá tiền hợp lý: Giá được tính dựa trên quãng đường và số hành khách, ưu tiên mức giá cạnh tranh.

5. Tính toán chi tiết lộ trình: Đối với các chuyến xe đã được sắp xếp, sử dụng API Goong để xác định quãng đường và thời gian dự kiến chi tiết giữa các điểm đón/trả của khách và lộ trình hiện có của chuyến xe.

Output: Danh sách các chuyến xe phù hợp, đã được sắp xếp theo độ ưu tiên, sẵn sàng hiển thị cho người dùng.

B, Thuật toán ghép tuyến

Sau khi người dùng chọn một chuyến xe, thuật toán này sẽ chịu trách nhiệm tích hợp yêu cầu đặt chỗ vào chuyến xe và tối ưu hóa lộ trình.

Input: Thông tin booking bao gồm customerId, tripId, điểm đón/trả, số hành khách và phương thức thanh toán.

Các bước thực hiện:

1. Kiểm tra đầu vào: Xác nhận tất cả thông tin bắt buộc (điểm đón/trả, phương thức thanh toán) đã được cung cấp. Nếu thiếu, giao dịch sẽ bị từ chối.
2. Khởi tạo giao dịch: Sử dụng MongoDB Transaction để đảm bảo tính nhất quán của dữ liệu. Mọi thao tác cập nhật (thêm booking, sửa đổi chuyến xe) sẽ diễn ra trong một giao dịch duy nhất, đảm bảo tính nguyên tử (atomic) của hoạt động.
3. Thêm điểm dừng: Thêm điểm đón và điểm trả của booking vào danh sách các điểm dừng (waypoints) của chuyến xe đã chọn. Mỗi điểm dừng mới sẽ được gắn kèm bookingId liên quan để dễ dàng quản lý và truy vấn.
4. Tối ưu lộ trình: Áp dụng thuật toán tham lam (Greedy Optimization) để sắp xếp lại thứ tự các điểm dừng trên lộ trình của chuyến xe. Thuật toán này hoạt động bằng cách iteratively chọn điểm đến tiếp theo là điểm gần nhất chưa được thăm, đồng thời đảm bảo điều kiện ràng buộc quan trọng: điểm đón của một booking phải luôn nằm trước điểm trả của cùng booking đó. Mục tiêu là tối thiểu hóa tổng quãng đường di chuyển.
5. Tính toán lại chi tiết lộ trình và giá: Gọi Goong Distance Matrix API để cập nhật chính xác:
    - Tổng quãng đường và thời gian di chuyển của toàn bộ chuyến xe sau khi thêm booking mới.

- Quãng đường, thời gian và giá tiền cụ thể cho booking vừa thêm (Giá = Giá cơ bản + Đơn giá × Quãng đường × Số hành khách).

6. Cập nhật dữ liệu: Lưu thông tin booking vào cơ sở dữ liệu và cập nhật dữ liệu của chuyến xe (thêm bookingId, cập nhật danh sách waypoints đã tối ưu, quãng đường và thời gian mới). Trạng thái của booking được đặt là "chờ xác nhận".
7. Kết thúc giao dịch: Commit transaction nếu tất cả các bước trên thành công. Trong trường hợp có bất kỳ lỗi nào xảy ra, transaction sẽ được rollback, đảm bảo dữ liệu không bị thay đổi một cách không nhất quán.

Output: Booking được ghép thành công, lộ trình chuyến xe được cập nhật và tối ưu.

2.7.3. Phân tích thuật toán

Độ phức tạp

- Thuật toán gợi ý tuyến: Độ phức tạp thời gian là O(NlogN), với N là số chuyến xe, chủ yếu do các thao tác lọc và sắp xếp. Việc sử dụng K-Means Clustering cũng đóng góp vào độ phức tạp nhưng thường được tối ưu cho các bộ dữ liệu vừa phải.
- Thuật toán ghép tuyến: Độ phức tạp thời gian là O(M2), với M là tổng số điểm dừng (waypoints) trên chuyến xe sau khi thêm booking, do việc sắp xếp lại lộ trình bằng thuật toán tham lam yêu cầu việc duyệt và so sánh các cặp điểm dừng.

Ưu điểm

- Hiệu quả với quy mô nhỏ: Thuật toán hoạt động hiệu quả và cung cấp kết quả nhanh chóng với số lượng điểm dừng nhỏ (thường dưới 15 điểm), phù hợp với quy mô hiện tại của hệ thống đặt xe sinh viên.
- Đảm bảo nhất quán dữ liệu: Việc sử dụng MongoDB Transaction là một điểm mạnh lớn, đảm bảo tính toàn vẹn và nhất quán của dữ liệu ngay cả khi có lỗi xảy ra trong quá trình ghép tuyến.
- Chính xác về thông tin địa lý: Tích hợp sâu với API Goong giúp đảm bảo độ chính xác cao trong việc tính toán quãng đường và thời gian thực tế.

Hạn chế

- Giới hạn về hiệu suất với số lượng điểm dừng lớn: Hiệu suất của thuật toán tham lam có thể suy giảm đáng kể khi số lượng điểm dừng trên một chuyến xe tăng lên quá lớn (ví dụ, trên 15-20 điểm), do bản chất O(M2).

- Không đảm bảo tối ưu toàn cục: Thuật toán tham lam, theo định nghĩa, không đảm bảo tìm được giải pháp tối ưu toàn cục (global optimum) cho bài toán tối ưu lộ trình. Điều này có nghĩa là có thể tồn tại một lộ trình tốt hơn mà thuật toán này không tìm thấy.

Hướng phát triển và cải tiến tương lai

Để nâng cao khả năng xử lý các trường hợp phức tạp hơn và tối ưu hóa hiệu suất cho quy mô lớn hơn, có thể xem xét các hướng phát triển sau:

- Áp dụng thuật toán tối ưu hóa phức tạp hơn: Thay thế hoặc bổ sung thuật toán tham lam bằng các thuật toán heuristic hoặc thuật toán tối ưu hóa tổ hợp tiên tiến hơn như A* Search hoặc các biến thể của Traveling Salesperson Problem (TSP) để đảm bảo tìm được lộ trình gần tối ưu hơn.
- Tối ưu hóa các yếu tố khác: Mở rộng tiêu chí tối ưu hóa để bao gồm cả thời gian chờ tại các điểm dừng, mức độ hài lòng của khách hàng, hoặc khả năng cân bằng tải giữa các chuyến xe.

Tổng kết chương 2

Chương 2 đã trình bày toàn diện quá trình **phân tích và thiết kế hệ thống vận tải hành khách ghép tuyến đa nền tảng**, đặt nền móng vững chắc cho các giai đoạn triển khai và phát triển tiếp theo.

Đầu tiên, chương này đã thực hiện **khảo sát và phân tích yêu cầu**, bao gồm việc xác định các yêu cầu chức năng và phi chức năng cần thiết để hệ thống hoạt động hiệu quả và đáp ứng nhu cầu thực tế của người dùng. Từ đó, các **trường hợp sử dụng (Use Case)** đã được xác định và **đặc tả chi tiết**, mô tả rõ ràng các chức năng chính cùng với luồng hoạt động, điều kiện tiên quyết, và kết quả mong muốn cho từng chức năng.

Tiếp theo, để minh họa luồng nghiệp vụ và tương tác giữa các tác nhân, các **biểu đồ hoạt động (Activity Diagram)** đã được xây dựng. Các biểu đồ này mô tả chi tiết các quy trình cốt lõi như yêu cầu chuyến đi, phân công tài xế, và kết thúc chuyến đi, giúp hình dung rõ ràng các bước thực hiện và sự phối hợp giữa Hành khách, Tài xế và Hệ thống.

Cuối cùng, chương đã đề xuất **kiến trúc tổng quan của hệ thống** dựa trên mô hình kiến trúc ba lớp (Client-Server, 3-tier Architecture), phác thảo cấu trúc tổng thể và mối quan hệ giữa các thành phần chính. Đồng thời, **thiết kế cơ sở dữ liệu** cũng được trình bày, bao gồm mô hình dữ liệu và cấu trúc các bảng (hoặc collection), đảm bảo việc lưu trữ và

quản lý dữ liệu hiệu quả, phục vụ cho toàn bộ các chức năng của hệ thống. Trình bày chi tiết **thuật toán gợi ý và ghép tuyến – một trong những thành phần quan trọng quyết định hiệu quả vận hành hệ thống.**

Những kết quả phân tích và thiết kế chi tiết trong chương này là kim chỉ nam vững chắc cho giai đoạn triển khai và phát triển hệ thống, được trình bày cụ thể trong Chương 3.

CHƯƠNG 3. TRIỂN KHAI, KIỂM THỬ VÀ ĐÁNH GIÁ HỆ THỐNG

Chương này trình bày chi tiết quá trình xây dựng các thành phần của hệ thống vận tải hành khách ghép tuyến đa nền tảng, các bước triển khai để đưa hệ thống vào hoạt động và các phương pháp kiểm thử được áp dụng để đảm bảo chất lượng.

3.1. Tổng quan xây dựng và công cụ phát triển

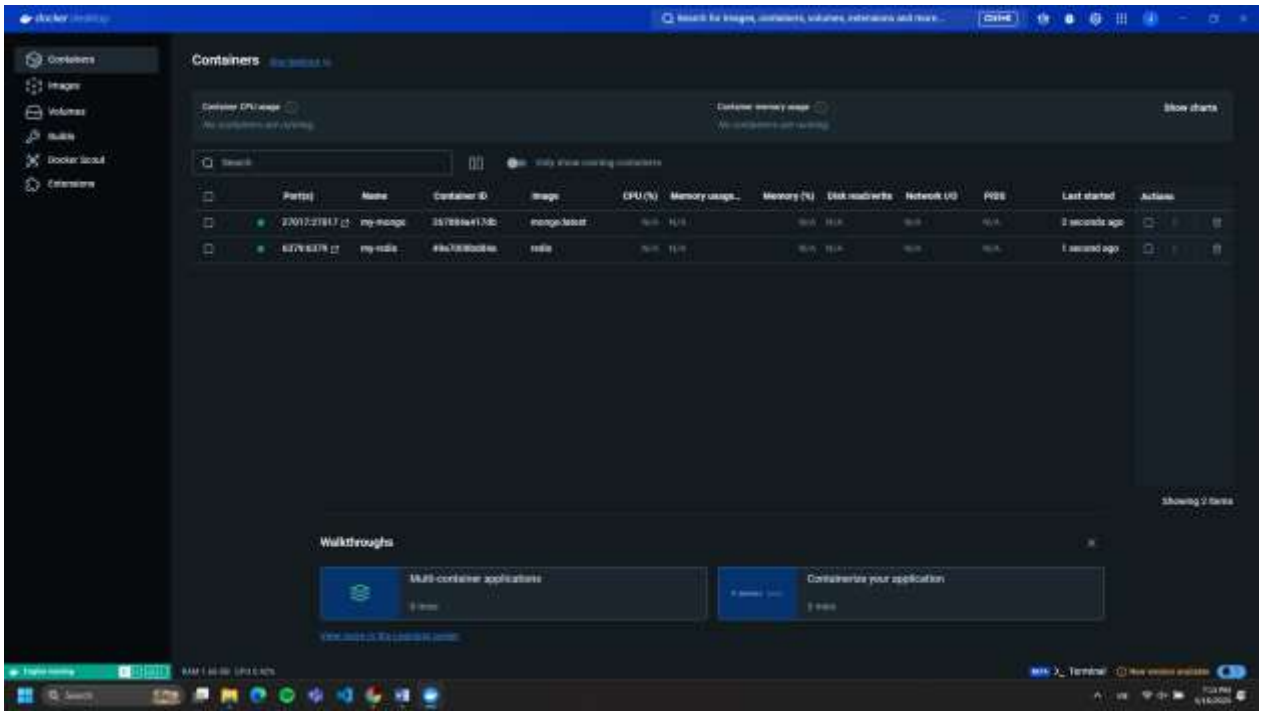
Hệ thống được triển khai dựa trên mô hình kiến trúc đa nền tảng, bao gồm ứng dụng di động (cho tài xế và hành khách), hệ thống backend xử lý nghiệp vụ, và giao diện Web Admin dành cho quản trị viên. Việc xây dựng hệ thống được chia thành ba phần chính: phát triển backend (API, xử lý logic), phát triển frontend (ứng dụng mobile và web), và triển khai toàn bộ hệ thống lên môi trường thực tế.

Để đảm bảo khả năng mở rộng, bảo trì và tương thích tốt, hệ thống sử dụng các công cụ hiện đại, phổ biến và dễ tích hợp trong môi trường phát triển. Các công cụ và mô tả cài đặt được trình trong bảng sau:

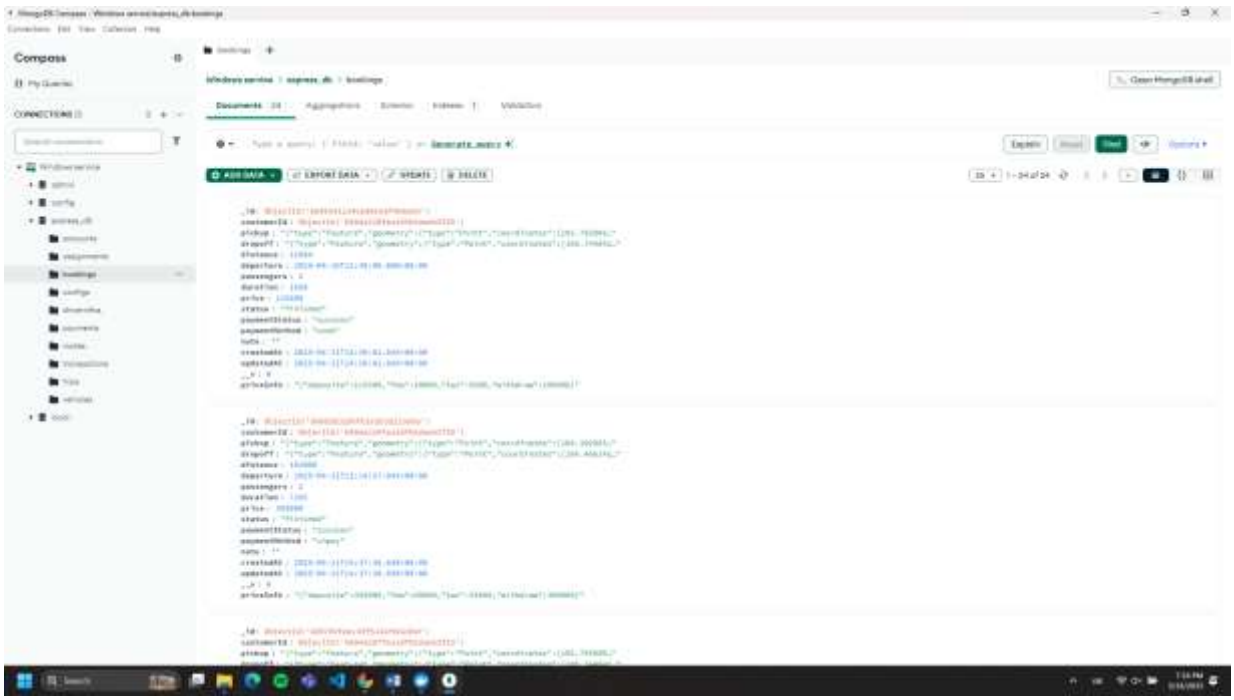
Bảng 23. Mô tả cài đặt công cụ hỗ trợ phát triển

Công cụ	Mục đích	Mô tả cài đặt
Visual Studio Code	Soạn thảo mã nguồn	Tải từ trang chủ https://code.visualstudio.com. Cài thêm các tiện ích: Prettier, ESLint, GitLens để hỗ trợ viết mã.
Node.js (>= 18)	Chạy môi trường backend	Tải từ https://nodejs.org. Tự động cài kèm npm. Dùng lệnh *node -v* và *npm -v* để kiểm tra.
Docker Desktop	Container hóa dịch vụ	Cài đặt từ https://www.docker.com. Sử dụng *Dockerfile* và *docker-compose.yml* để khởi tạo backend, MongoDB, Redis đồng bộ.
MongoDB (qua Docker)	Cơ sở dữ liệu NoSQL	Được cài đặt và cấu hình tự động thông qua *docker-compose.yml*
Redis (qua Docker)	Cache dữ liệu, lưu trữ thông tin	Được cài đặt và cấu hình tự động thông qua *docker-compose.yml.*
Expo CLI	Build và test ứng dụng React Native	Cài bằng lệnh *npm install -g expo-cli*. Chạy ứng dụng bằng *npx expo start*. Cài thêm ứng dụng Expo Go trên thiết bị hoặc giả lập để kiểm tra.

MongoDB Compass	Quản lý và trực quan hóa dữ liệu MongoDB	Tải từ https://www.mongodb.com/products/tools/compass. Kết nối tới cơ sở dữ liệu qua mạng nội bộ hoặc URI.
Postman	Gửi và kiểm thử API	Dùng trực tiếp hoặc tải về từ https://www.postman.com/
Firebase	Lưu ảnh, gửi OTP, thông báo	Dùng trên trình duyệt tại https://console.firebase.google.com/



Hình 13. Giao diện Docker Desktop khi chạy MongoDB và Redis



Hình 14. Kết nối MongoDB Compass vào cơ sở dữ liệu

3.2. Xây dựng hệ thống

3.2.1. Xây dựng server

Backend là thành phần nòng cốt của hệ thống, chịu trách nhiệm xử lý các yêu cầu nghiệp vụ, kết nối với cơ sở dữ liệu và phản hồi dữ liệu cho frontend. Hệ thống backend được xây dựng bằng **Node.js** kết hợp với **TypeScript**, áp dụng kiến trúc **MVC (Model – View – Controller)** nhằm phân tách rõ ràng giữa các tầng xử lý. Ngoài ra, toàn bộ ứng dụng được container hóa bằng **Docker**, giúp đảm bảo tính đồng nhất trên các môi trường phát triển, kiểm thử và triển khai thực tế.

3.2.1.1 Khởi tạo và cấu hình dự án

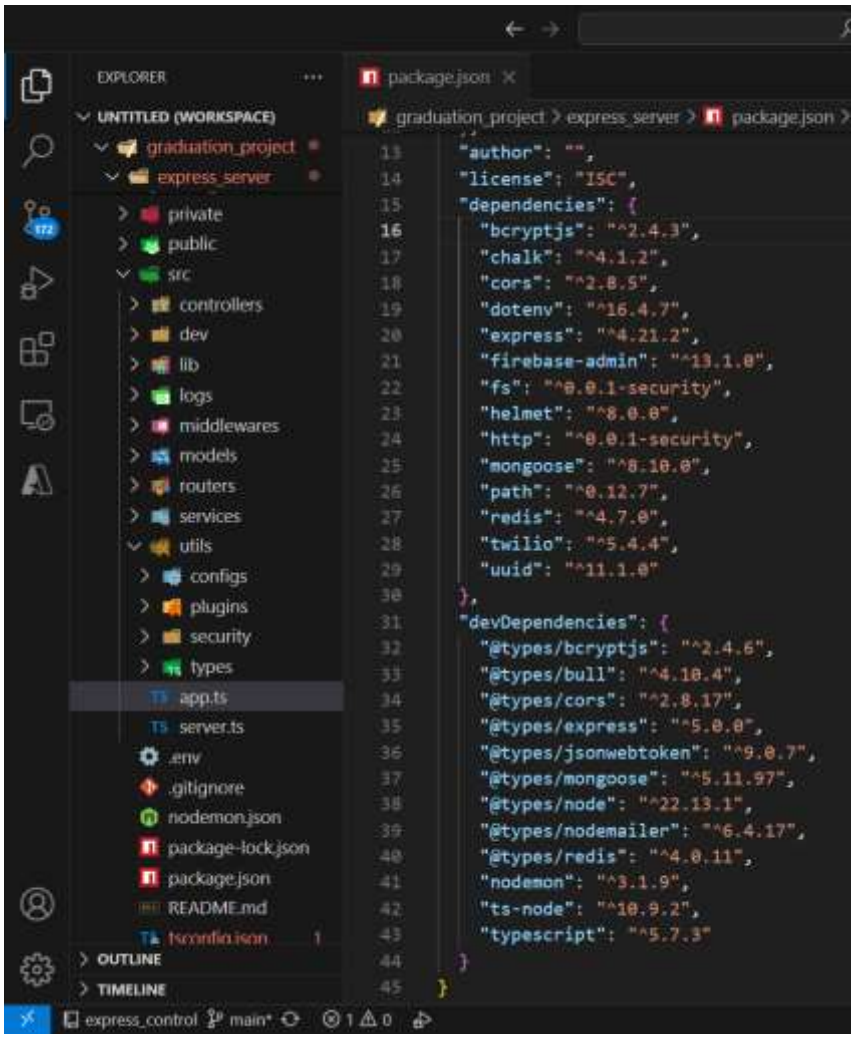
Quá trình khởi tạo dự án bắt đầu với lệnh *"npm init –y"*, tạo ra tệp *package.json* với các thiết lập mặc định. Sau đó, các thư viện được cài đặt thông qua:

- *"npm install"*: Cài đặt các thư viện phục vụ chạy chính thức (express, mongoose, redis, jsonwebtoken,...).
- *"npm install –D"*: Cài đặt các thư viện dùng cho phát triển (typescript, ts-node-dev, @types,...).

Các tệp cấu hình quan trọng bao gồm:

- **tsconfig.json**: Thiết lập môi trường TypeScript (target ES2020, module CommonJS, outDir là dist, rootDir là src...).
- **.env**: Quản lý biến môi trường như URI kết nối MongoDB, JWT secret, Redis host.
- **nodemon.json**: Giúp tự động khởi động lại server khi mã nguồn thay đổi.
- **.gitignore**: Loại trừ các tệp như node_modules, dist, .env khỏi hệ thống quản lý phiên bản.

3.2.1.2 Cấu trúc mã nguồn



Hình 15. Cấu trúc mã nguồn và các gói thư viện của backend

Mã nguồn backend được tổ chức rõ ràng, module hóa và theo mô hình MVC. Cấu trúc thư mục chính bao gồm:

- **controllers/:** Xử lý logic khi route được gọi, trả về JSON (thay cho View trong MVC truyền thống).
- **services/:** Chứa các nghiệp vụ chính như đặt chuyến, phân công tài xế, tính giá,...

- **models/:** Định nghĩa schema MongoDB bằng Mongoose, tương ứng với các collection.
- **routes/:** Khai báo endpoint API và ánh xạ tới controller phù hợp.
- **middlewares/:** Xử lý các chức năng trung gian như xác thực, log, kiểm tra lỗi, validate dữ liệu.
- **utils/:** Hàm tiện ích dùng chung như tạo JWT, tương tác Redis, Firebase,...
- **configs/:** Cấu hình kết nối MongoDB, Redis, và biến môi trường.
- **types/:** Các kiểu dữ liệu TypeScript tuỳ chỉnh dùng xuyên suốt dự án.
- **app.ts:** Khởi tạo Express app, cấu hình middleware, CORS, router.
- **server.ts:** Tệp khởi chạy chính, lắng nghe kết nối tới server.

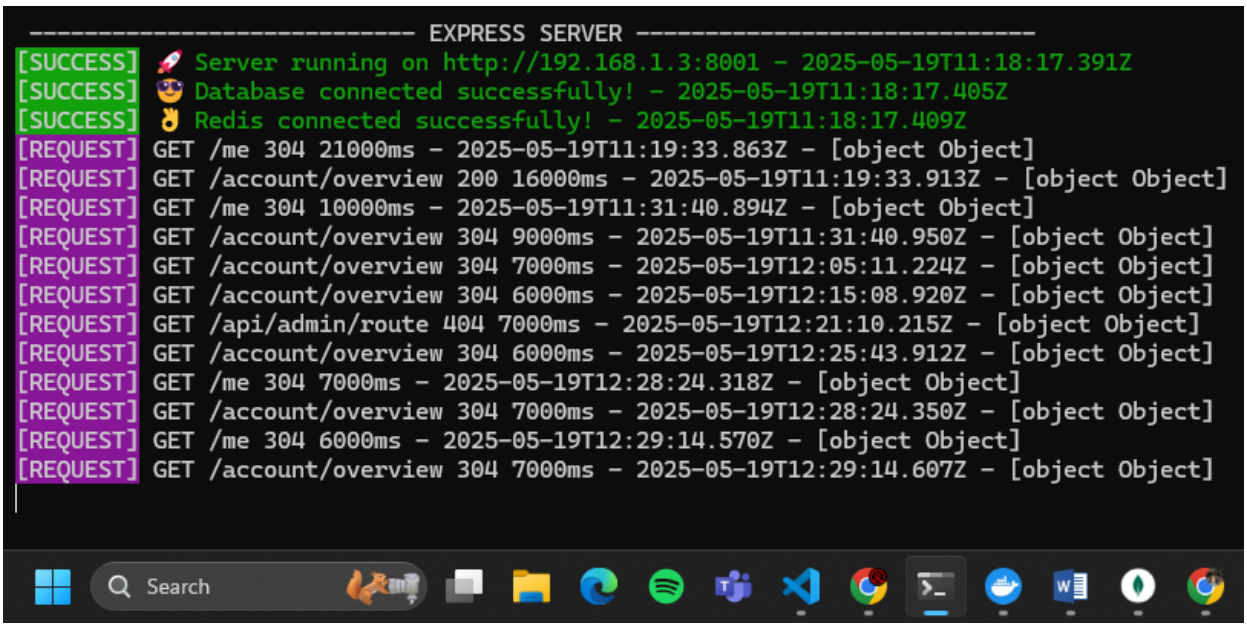
3.2.1.3 Giao tiếp RESTful API

Hệ thống backend sử dụng kiến trúc **RESTful API** để giao tiếp giữa client và server. Mỗi tài nguyên được định danh thông qua URI, và được thao tác bằng các phương thức HTTP tiêu chuẩn như **GET**, **POST**, **PUT**, **DELETE**. Đặc điểm nổi bật của REST là tính **stateless**, tức là server không lưu trạng thái giữa các yêu cầu. Dữ liệu trao đổi chủ yếu dưới định dạng **JSON**, đảm bảo khả năng phân tích, hiển thị và xử lý hiệu quả ở phía client. Việc triển khai các endpoint được xác thực bằng log terminal, thể hiện khả năng tiếp nhận và phản hồi của hệ thống một cách trơn tru và đáng tin cậy.

3.2.1.4 Middleware và luồng xử lý

Để tối ưu hóa luồng xử lý API, hệ thống sử dụng các middleware tách riêng theo từng chức năng. Điều này giúp mã nguồn rõ ràng, dễ bảo trì và mở rộng:

- **asyncHandler.ts:** Xử lý các hàm bất đồng bộ, tránh callback hell.
- **auth.ts:** Xác thực người dùng bằng JWT, bảo vệ tài nguyên nội bộ.
- **errorHandler.ts:** Chuẩn hóa phản hồi lỗi, hỗ trợ gỡ lỗi nhanh.
- **logRequest.ts:** Ghi log chi tiết các request để giám sát hoạt động.
- **validate.ts:** Kiểm tra và xác thực dữ liệu đầu vào từ client.

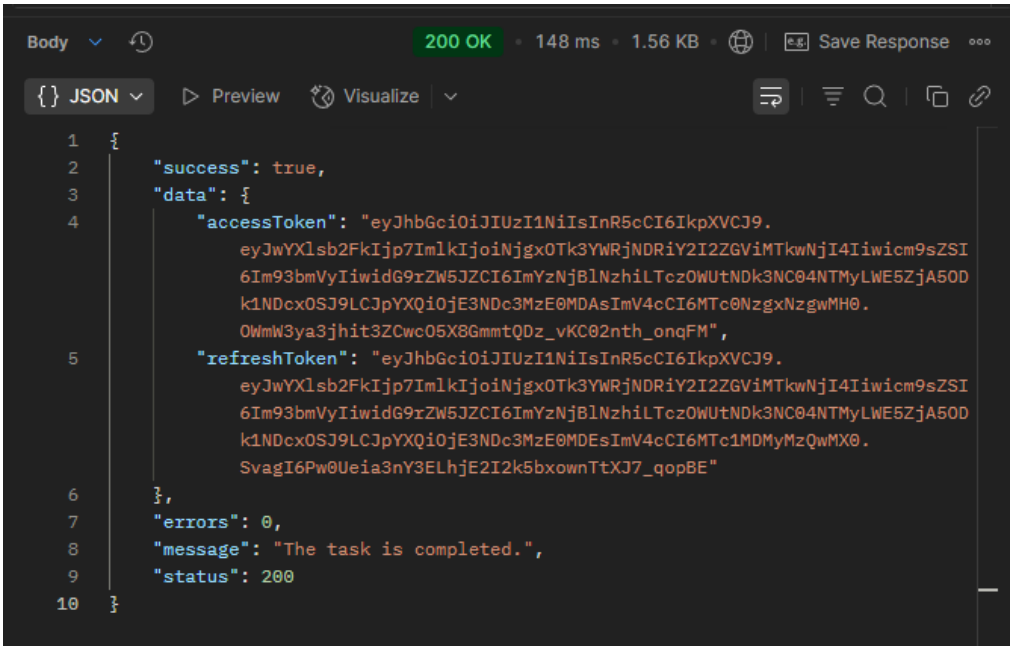


Hình 16. Terminal ghi log chi tiết các request

3.2.1.5 *Chuẩn hóa phản hồi API*

Tất cả phản hồi từ server được chuẩn hóa theo một cấu trúc nhất quán, giúp frontend dễ xử lý và debug thuận tiện:

- success: (boolean) Cho biết yêu cầu thành công hay thất bại.
- status: (number – optional) Mã HTTP trả về.
- data: (any) Dữ liệu phản hồi chính.
- message: (string) Mô tả ngắn về kết quả xử lý.
- errors: (object – optional) Chi tiết lỗi nếu có.



Hình 17. Cấu trúc response nhất quán

3.2.1.6 Cơ chế Bảo mật Hệ thống

Hệ thống được thiết kế với nhiều lớp bảo mật cơ bản nhằm đảm bảo an toàn dữ liệu, xác thực người dùng và phòng chống các mối đe dọa phổ biến:

- **Xác thực người dùng:** Sử dụng JWT với hai loại token (Access và Refresh Token), trong đó Refresh Token được lưu trữ và quản lý trong Redis theo thiết bị, hỗ trợ thu hồi linh hoạt.
- **Phân quyền truy cập:** Áp dụng mô hình kiểm soát truy cập theo vai trò người dùng, kiểm tra vai trò người dùng qua middleware để giới hạn quyền truy cập phù hợp.
- **Mã hóa dữ liệu:** Mật khẩu người dùng được mã hóa bằng bcrypt; các thông tin nhạy cảm khác được mã hóa khi cần thiết.
- **Kiểm soát truy cập & phòng chống tấn công:** cấu hình CORS, giám sát log hành vi bất thường.
- **Bảo vệ dữ liệu đầu vào:** Sử dụng Zod để xác thực dữ liệu đầu vào và Helmet để thiết lập các header bảo mật, chống lại các lỗ hổng như XSS, Clickjacking.
- **Ngăn chặn XSS:** Kết hợp các biện pháp như sanitization, encoding và cấu hình CSP nhằm bảo vệ khỏi các đoạn mã độc được chèn từ phía người dùng.
- **Chữ ký điện tử trong thanh toán:** Áp dụng ký số với thư viện crypto nhằm đảm bảo tính toàn vẹn, xác thực và không chối bỏ trong giao dịch thanh toán.

3.2.1.7 Đánh giá tổng kết

Phần backend được triển khai thành công với kiến trúc rõ ràng, chuẩn hóa toàn bộ phản hồi, xử lý bất đồng bộ hiệu quả và đảm bảo bảo mật qua JWT. Cấu trúc module hóa giúp dễ dàng phát triển, kiểm thử và mở rộng trong tương lai. Đây là nền tảng vững chắc cho các tính năng nâng cao như thanh toán online, thống kê nâng cao và tích hợp dịch vụ bên thứ ba.

3.2.2. Xây dựng website

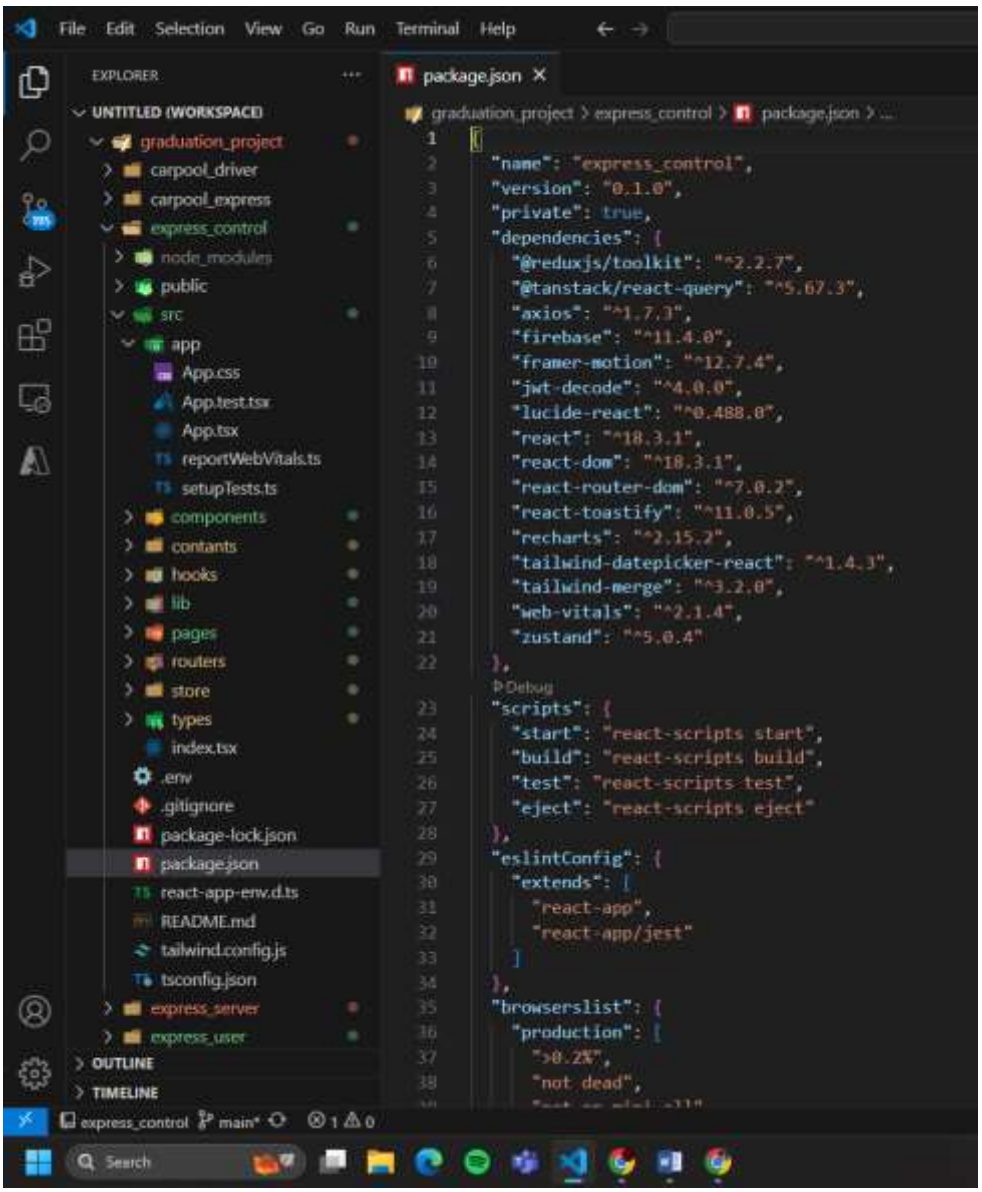
Web Admin đóng vai trò là công cụ quản lý chính cho hệ thống, cho phép quản trị viên theo dõi, vận hành và cấu hình hệ thống xe ghép tuyến. Giao diện được xây dựng bằng React và TypeScript với kiến trúc được tổ chức theo mô hình **component-based**, với mỗi chức năng lớn được tách thành một module riêng biệt như: Quản lý tài khoản, Quản lý

tuyến, Báo cáo thống kê và Cấu hình hệ thống. Layout chính bao gồm **Sidebar, Header và khu vực Content**, đảm bảo tính nhất quán và dễ điều hướng trên toàn bộ hệ thống.

3.2.2.1 Cấu trúc mã nguồn

Mã nguồn frontend được tổ chức rõ ràng, thuận tiện cho phát triển và mở rộng:

- **components/:** Chứa các thành phần giao diện tái sử dụng như nút bấm, bảng dữ liệu, modal, layout,... giúp tăng khả năng tái sử dụng và giữ tính nhất quán về giao diện.
- **pages/:** Tổ chức theo từng chức năng lớn (Dashboard, Users, Routes, Settings).
- **constants/:** Định nghĩa các hằng số dùng chung trong toàn bộ ứng dụng như tên vai trò, thông điệp hệ thống, màu sắc,...
- **lib/:** Định nghĩa các hàm cấu hình (Sử dụng Axios và interceptor để tự động gắn token, ...), hàm dùng chung (utils) với khả năng tái sử dụng cao.
- **stores/:** Sử dụng Zustand để quản lý trạng thái cục bộ (người dùng, menu, v.v.).
- **hooks/:** Custom hook như useAuth, useUser, useRoutes,...
- **routes/:** Định nghĩa cấu trúc router, kết hợp phân quyền theo vai trò.
- **types/:** Khai báo các kiểu dữ liệu TypeScript dùng chung, giúp tăng độ an toàn khi phát triển và giảm lỗi.
- **app/:** Chứa các thành phần khởi tạo ứng dụng như cấu hình router, layout, theme, query client,... đóng vai trò là lớp gốc của toàn bộ hệ thống.
- **index.tsx:** Tệp khởi tạo chính của ứng dụng, nơi gắn ứng dụng React vào DOM và khởi động toàn bộ hệ thống.



Hình 18. Cấu trúc mã nguồn và các gói thư viện của website

3.2.2.2 Giao tiếp với Backend

Tầng giao tiếp giữa frontend và backend được thực hiện thông qua các **RESTful API**, sử dụng thư viện **Axios**. Toàn bộ request được tự động đính kèm token xác thực thông qua **interceptor**, giúp đảm bảo bảo mật và đơn giản hóa việc gọi API. Ngoài ra, **React Query** được sử dụng để quản lý dữ liệu bất đồng bộ, hỗ trợ cache, tự động làm mới dữ liệu khi cần thiết, và giảm thiểu số lần gọi lại không cần thiết tới server.

3.2.2.3 Các chức năng chính

Website hiện tại phục vụ vai trò là **hệ thống quản trị**, tập trung vào các chức năng:

- **Quản lý tài khoản**: Xem danh sách người dùng, phân loại khách hàng và tài xế, hiển thị chi tiết tài khoản và hồ sơ tài xế.

- **Quản lý tuyến**: Tạo tuyến đường, cập nhật thông tin, phân công tài xế theo tuyến.
- **Thống kê – báo cáo**: Xem số lượt đặt xe, doanh thu, trạng thái các chuyến.
- **Cấu hình hệ thống**: Quản lý giá cước, các khoản phí hoặc thông số cấu hình chung.

Toàn bộ các bảng dữ liệu đều hỗ trợ **tìm kiếm, phân trang và lọc theo trạng thái,** giúp dễ dàng truy xuất và xử lý khối lượng dữ liệu lớn.

3.2.2.4 Tối ưu hóa và trải nghiệm người dùng

- **Zustand** được dùng để lưu trữ trạng thái toàn cục (như thông tin đăng nhập), thay cho Context, giúp giảm độ phức tạp và cải thiện hiệu suất.
- **Framer Motion** được áp dụng cho các hiệu ứng chuyển trang, modal hoặc mở rộng menu, mang lại trải nghiệm mượt mà hơn.
- Giao diện được thiết kế **responsive**, tương thích tốt với nhiều độ phân giải, đảm bảo khả năng sử dụng trên các thiết bị khác nhau.

3.2.2.5 Đánh giá tổng kết

Website được triển khai với kiến trúc rõ ràng, chia module hợp lý, đảm bảo khả năng mở rộng và bảo trì trong tương lai. Việc kết hợp các thư viện hiện đại như React Query, Zustand giúp tối ưu hiệu suất hiển thị, khả năng tương tác và giữ mã nguồn gọn gàng. Đây là nền tảng vững chắc cho việc mở rộng sang mobile app hoặc tích hợp các tính năng nâng cao như thông báo real-time, biểu đồ phân tích dữ liệu.

3.2.3. Xây dựng ứng dụng di động

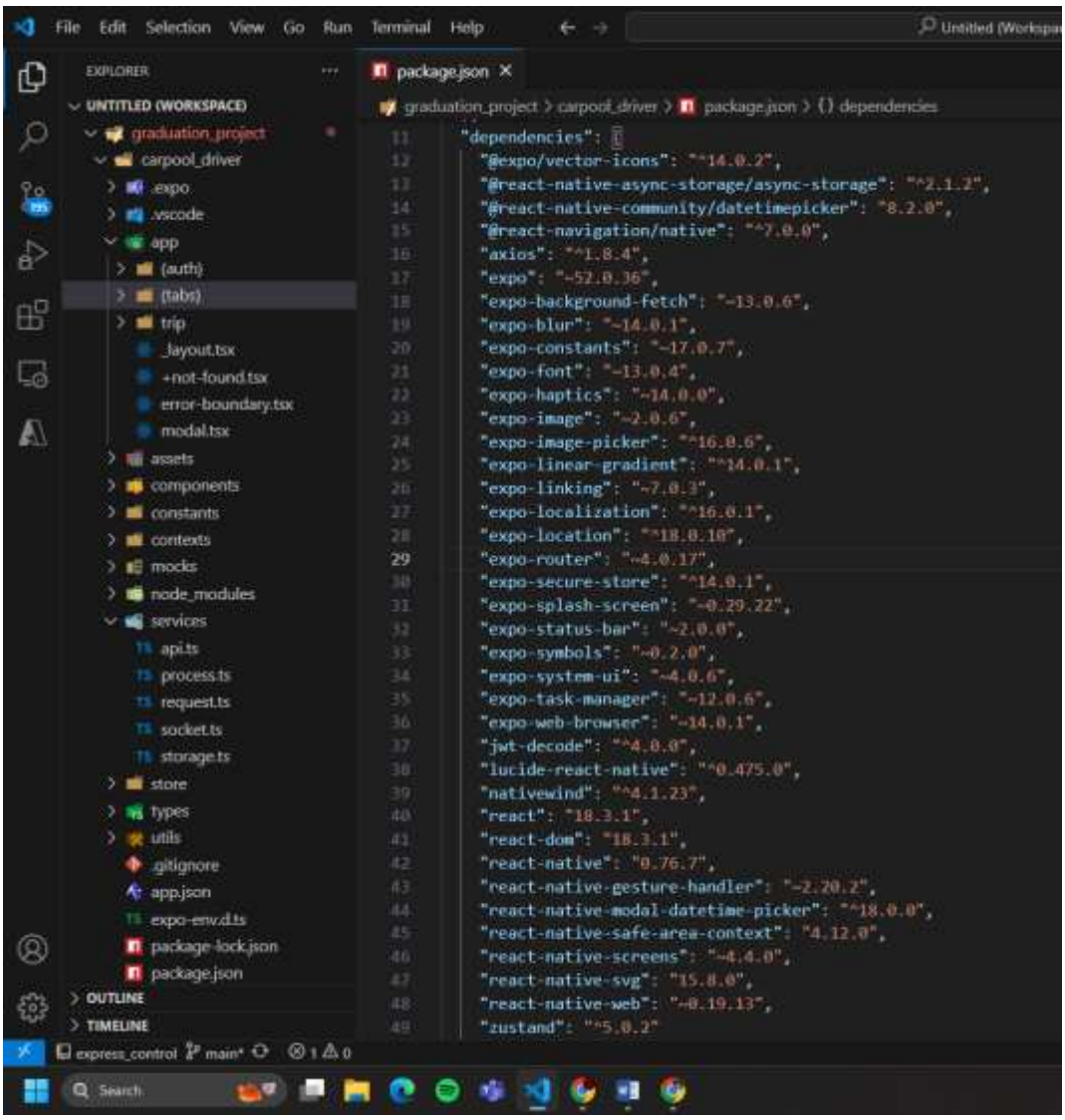
Ứng dụng di động phục vụ người dùng đầu cuối (khách hàng và tài xế), được xây dựng bằng **React Native** kết hợp với **TypeScript** nhằm đảm bảo hiệu suất, độ ổn định và khả năng mở rộng. Ứng dụng sử dụng **Expo** để đơn giản hóa quy trình phát triển, thử nghiệm và phân phối.

Kiến trúc được tổ chức theo mô hình **modular – component-based**, với các màn hình chính chia thành hai phân hệ: **người dùng** (đặt xe, xem chuyến, thanh toán) và **tài xế** (nhận chuyến, theo dõi tuyến, xác nhận hoàn thành).

3.2.3.1 Cấu trúc mã nguồn

Mã nguồn ứng dụng được tổ chức rõ ràng theo mô hình module hóa và định tuyến dựa trên thư mục, phù hợp với cấu trúc của Expo Router v3. Điều này giúp dễ dàng phát triển, mở rộng và bảo trì về sau. Dưới đây là mô tả các thư mục chính trong dự án:

- **app/:** Chứa các màn hình chính của ứng dụng, tổ chức theo routing cấu trúc thư mục. Mỗi thư mục con như auth/, trip/ tương ứng với một nhóm tính năng. Ngoài ra có các tệp đặc biệt như:
  - **_layout.tsx:** Layout mặc định áp dụng cho toàn bộ app.
  - **not-found.tsx, error-boundary.tsx:** Xử lý lỗi và điều hướng trang không tồn tại.
  - **modal.tsx:** Giao diện modal dùng chung toàn app.
- **assets/:** Chứa các tài nguyên tĩnh như hình ảnh, biểu tượng, fonts,...
- **components/:** Các thành phần giao diện tái sử dụng như nút, ô nhập, thẻ hiển thị, modal,... được chuẩn hóa để giữ tính thống nhất giao diện.
- **contexts/:** Cung cấp ngữ cảnh toàn cục (React Context) cho các logic dùng chung như thông tin người dùng, xác thực, cấu hình app,...
- **lib/:** Chứa các hàm tiện ích hỗ trợ gọi API bằng Axios. Đã cấu hình sẵn interceptor để tự động đính kèm access token và xử lý lỗi toàn cục.
- **services/:** Định nghĩa các hàm tương tác với backend (login, đặt chuyến, cập nhật trạng thái, v.v.) phân chia theo từng domain nghiệp vụ.
- **store/:** Sử dụng Zustand để quản lý trạng thái cục bộ như trạng thái đăng nhập, thông tin người dùng, menu,...
- **types/:** Định nghĩa các kiểu dữ liệu dùng chung trong toàn bộ dự án, giúp tăng tính ổn định và an toàn kiểu khi phát triển với TypeScript.
- **utils/:** Chứa các hàm tiện ích như định dạng thời gian, tạo mã định danh, kiểm tra quyền truy cập,...
- **app.config.ts:** Tệp cấu hình ứng dụng Expo, khai báo tên app, icon, splash screen, deep link, biến môi trường,...
- **index.ts:** Điểm khởi tạo chính của ứng dụng, dùng để gắn ứng dụng vào AppRegistry và khởi chạy toàn bộ hệ thống.



Hình 19. Cấu trúc mã nguồn và các gói thư viện của mobile app

3.2.3.2 Các chức năng chính

Ứng dụng mobile bao gồm hai phân hệ chính:

A. Khách hàng:

- Đăng ký, đăng nhập, cập nhật hồ sơ.
- Đặt chuyến theo tuyến có sẵn.
- Xem thông tin chuyến sắp tới và lịch sử chuyến.
- Đánh giá tài xế sau khi hoàn thành chuyến đi.
- Nhận thông báo khi có thay đổi (tuyến bị huỷ, cập nhật thời gian,...).

B. Tài xế:

- Đăng nhập, xác minh hồ sơ.
- Nhận danh sách các tuyến được phân công.

- Bắt đầu – kết thúc chuyến, cập nhật trạng thái thực tế.
- Xác nhận hành khách và điểm dừng.
- Xem lịch làm việc và doanh thu tổng hợp.

3.2.3.3 Giao tiếp với Backend

Tương tự website, ứng dụng mobile sử dụng **Axios với interceptor** để giao tiếp với backend thông qua RESTful API. Tất cả các request đều được gắn kèm token JWT, đảm bảo an toàn và xác thực người dùng ở từng phiên truy cập.

Việc quản lý dữ liệu bất đồng bộ sử dụng **React Query**, kết hợp với **Zustand** cho trạng thái cục bộ, giúp đảm bảo tốc độ phản hồi nhanh và trải nghiệm mượt mà.

3.2.3.4 Tối ưu và trải nghiệm người dùng

Ứng dụng được tối ưu hóa để đảm bảo hiệu suất cao và trải nghiệm người dùng (UX) mượt mà, với giao diện đơn giản, dễ sử dụng. Các cải tiến bao gồm:

- **Lazy Loading**: Tải các thành phần như hình ảnh, bản đồ theo nhu cầu, giảm thời gian tải và tiết kiệm tài nguyên.
- **Memoization**: Sử dụng React.memo và useMemo để ngăn render không cần thiết, tăng hiệu suất khi xử lý dữ liệu lớn.
- **Caching dữ liệu**: React Query lưu trữ dữ liệu API, giảm yêu cầu server và hỗ trợ truy cập ngoại tuyến tạm thời.
- **Push Notification**: Tích hợp Firebase Cloud Messaging (FCM) để gửi thông báo real-time như trạng thái chuyến,...
- **Điều hướng**: React Navigation tách biệt luồng cho khách hàng và tài xế.
- **Giao diện**: hỗ trợ sáng/tối, đa ngôn ngữ (i18n), và phản hồi xúc giác.
- **Tích hợp bản đồ**: Goong API/Google Maps SDK để hiển thị bản đồ, gợi ý địa điểm, và theo dõi lộ trình. Xử lý GPS thời gian thực với Socket.io.

3.2.3.5 Đánh giá tổng kết

Ứng dụng mobile đóng vai trò quan trọng trong hệ thống đặt xe, cung cấp trải nghiệm nhanh gọn và trực tiếp đến người dùng cuối. Kiến trúc rõ ràng, tách biệt vai trò, cùng với công nghệ hiện đại như React Native, Zustand và React Query giúp đảm bảo hiệu

suất và dễ dàng mở rộng trong tương lai. Ứng dụng có thể được phân phối trên cả Android và iOS qua Expo hoặc build thủ công bằng EAS Build.

3.3. Triển khai hệ thống

3.3.1. Triển khai server

Hệ thống backend sau khi hoàn thiện được triển khai lên Google Cloud Platform (GCP) để đảm bảo khả năng truy cập từ internet và phục vụ các ứng dụng frontend (Web Admin, Rider App, Driver App) một cách ổn định.

Việc triển khai được thực hiện qua các bước chính sau:

- Tạo máy ảo thông qua Google Compute Engine, sử dụng hệ điều hành Ubuntu.
- Cài đặt môi trường Node.js, Docker và các công cụ hỗ trợ như PM2 để quản lý tiến trình chạy server.
- Sử dụng Docker Compose để dựng các dịch vụ cần thiết như MongoDB (replica set), Redis và chính server backend.
- Cấu hình mở cổng HTTP/HTTPS và thiết lập firewall cho máy ảo.
- Quản lý domain và cấu hình reverse proxy bằng Nginx, giúp định tuyến các yêu cầu tới đúng cổng dịch vụ.
- Kiểm tra kết nối từ frontend đến backend để đảm bảo hệ thống hoạt động ổn định trên môi trường triển khai thực tế.

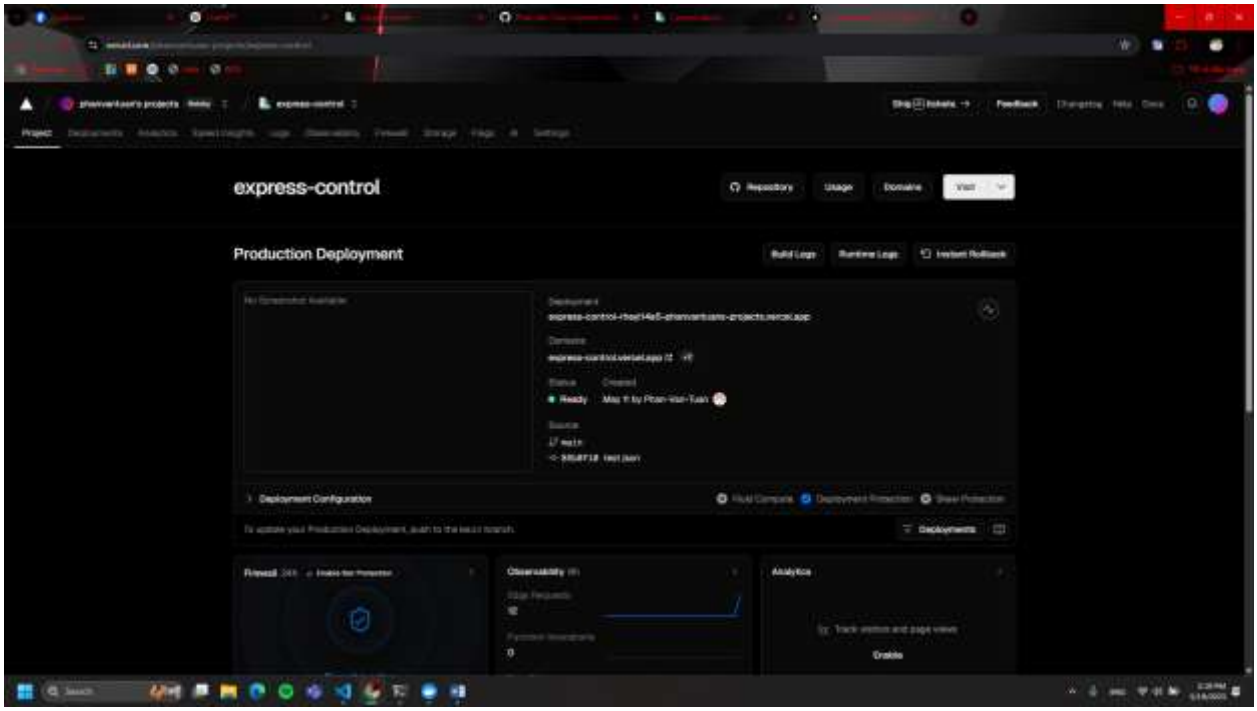
3.3.2. Triển khai website

Phần giao diện quản trị (Web Admin) được triển khai trên nền tảng Vercel – một dịch vụ đám mây tối ưu cho các ứng dụng frontend hiện đại, đặc biệt là các dự án sử dụng React, Next.js, hoặc các framework dựa trên JavaScript.

Quy trình triển khai Web Admin gồm các bước chính:

- Đồng bộ mã nguồn từ GitHub lên Vercel thông qua kết nối tài khoản.
- Cấu hình thông số môi trường (environment variables), đảm bảo khả năng tương tác với backend.
- Cấu hình tự động build và deploy khi có thay đổi trên nhánh chính (main).
- Thiết lập custom domain (nếu cần) và bật tính năng HTTPS để đảm bảo bảo mật khi truy cập.

Việc sử dụng Vercel giúp rút ngắn thời gian triển khai, hỗ trợ CI/CD tự động, dễ theo dõi lịch sử phiên bản và cung cấp hiệu suất cao với cơ chế tối ưu phân phối nội dung (CDN). Sau khi triển khai thành công, project sẽ hiển thị trạng thái thành công như được biểu thị ở hình dưới:



Hình 20. Mã nguồn từ github được triển khai trên vercel

3.3.3. Xuất bản ứng dụng di động

Hai ứng dụng di động gồm Rider App (cho khách hàng) và Driver App (cho tài xế) được phát triển bằng React Native kết hợp với Expo, cho phép xây dựng ứng dụng đa nền tảng từ một cơ sở mã nguồn duy nhất. Trong quá trình phát triển, em sử dụng Expo Application Services (EAS) để thực hiện việc build ứng dụng và triển khai thử nghiệm.

Tuy nhiên, do điều kiện thực tế không có tài khoản Google Play Console và Apple Developer Program, em không thể xuất bản ứng dụng lên kho ứng dụng chính thức (Google Play và App Store). Thay vào đó, em lựa chọn phương án build ứng dụng ở dạng cục bộ để cài đặt và kiểm thử trực tiếp trên thiết bị thật.

Cụ thể:

Đối với Android:

- Sử dụng Expo Application Services (EAS) để build app bằng lệnh: *"eas build -p android --profile preview"*

- Tải file .apk từ Expo và cài đặt trực tiếp lên thiết bị Android bằng cách bật quyền "Cài đặt ứng dụng từ nguồn không xác định".

Đối với iOS:

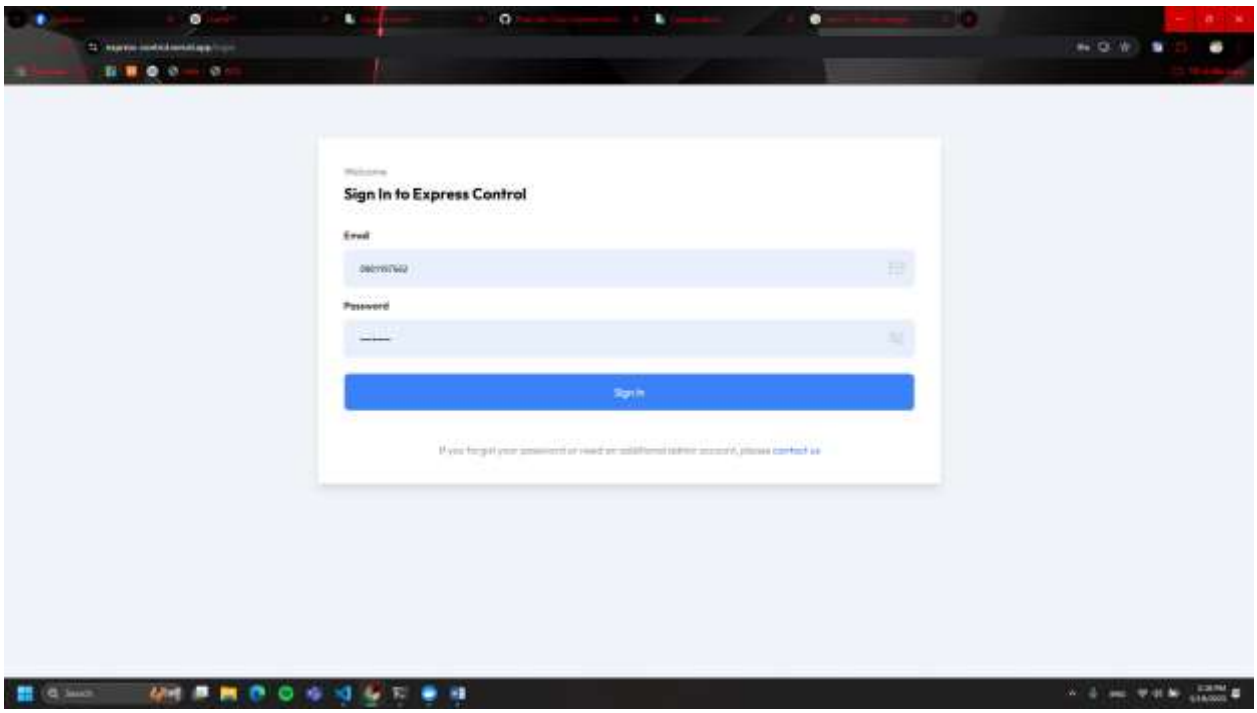
- Do hệ điều hành iOS không cho phép cài đặt .ipa từ ngoài App Store nếu không có tài khoản nhà phát triển, em sử dụng Expo Go để kiểm thử trực tiếp ứng dụng qua mã QR.
- Việc chạy thử trên iOS thông qua Expo Go vẫn đảm bảo hiển thị giao diện, tương tác và logic nghiệp vụ tương tự bản cài đặt thực tế, tuy một số tính năng native bị giới hạn.

Giải pháp build cục bộ kết hợp với kiểm thử qua Expo Go giúp em đánh giá đầy đủ các chức năng và trải nghiệm người dùng trong môi trường thực tế, đồng thời tiết kiệm chi phí đáng kể trong giai đoạn phát triển ban đầu. Hình thức này phù hợp cho thử nghiệm nội bộ trước khi phát hành chính thức trong tương lai.

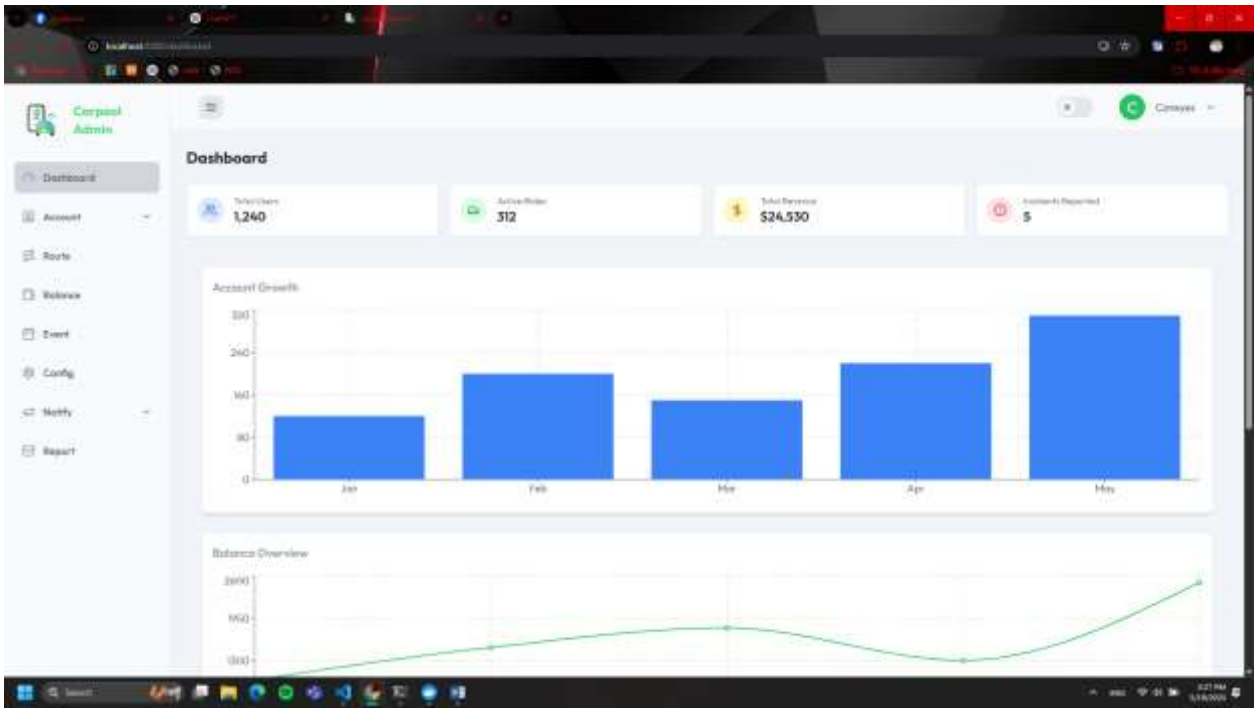
3.4. Một số kết quả đạt được

Sau quá trình xây dựng và triển khai, hệ thống đã hoàn thiện các thành phần chính bao gồm Website quản trị (Admin Dashboard) và hai ứng dụng di động dành cho người dùng (Rider App) và tài xế (Driver App). Các chức năng cốt lõi như xác thực, quản lý tài khoản, đặt xe, theo dõi chuyến đi, xét duyệt tài xế, thống kê tài chính... đã được tích hợp đầy đủ và vận hành ổn định.

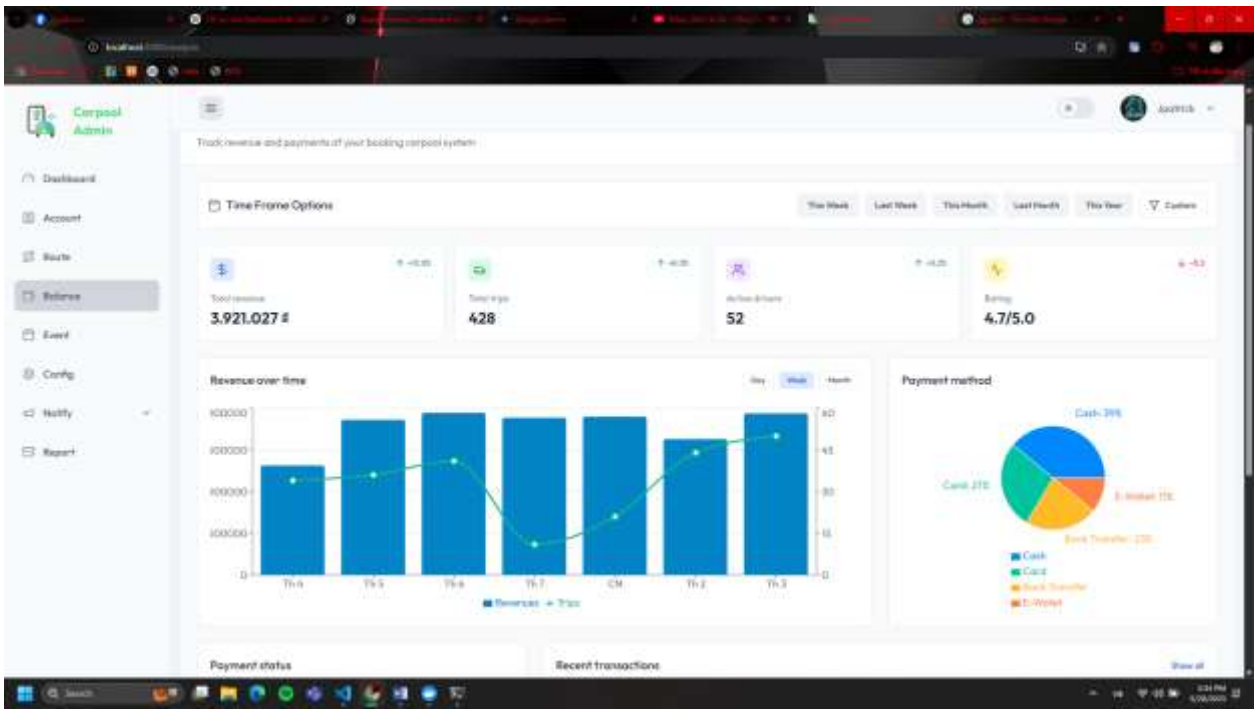
Dưới đây là một số giao diện tiêu biểu thể hiện kết quả đạt được trong quá trình thực hiện đồ án:



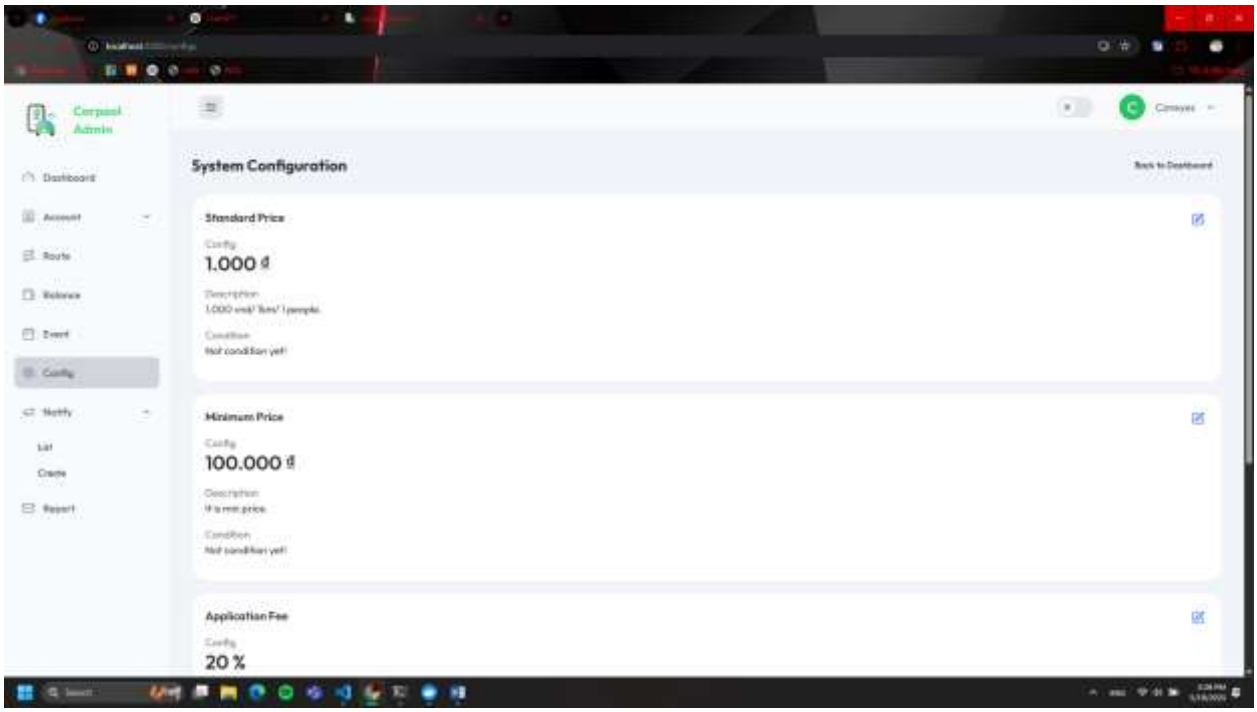
Hình 21. Màn đăng nhập trên website



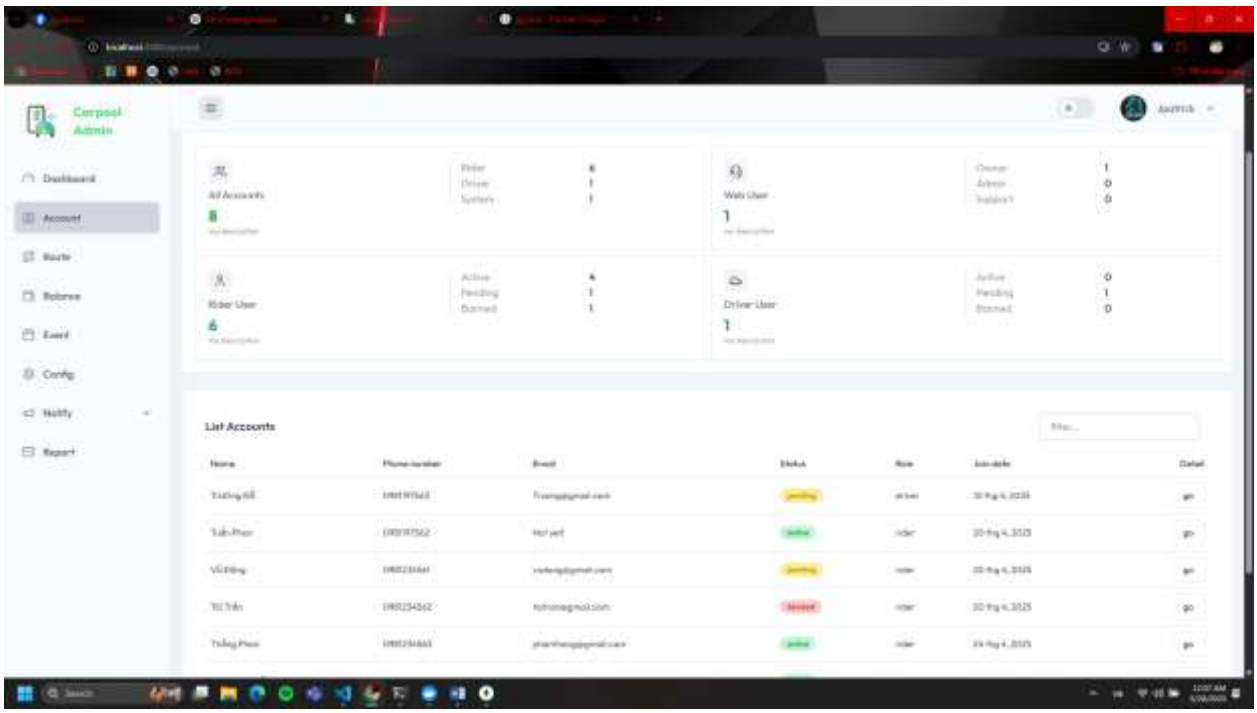
Hình 22. Màn hình chính trên website



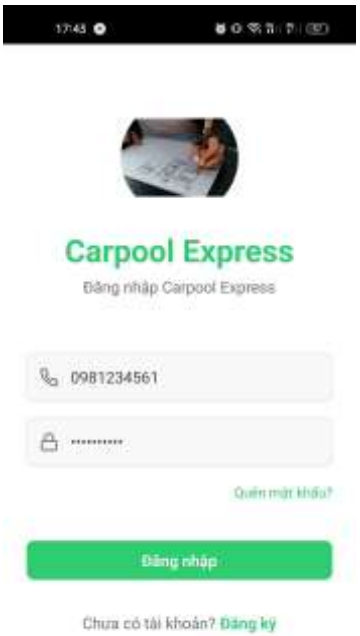
Hình 23. Màn quản lý tài chính trên website



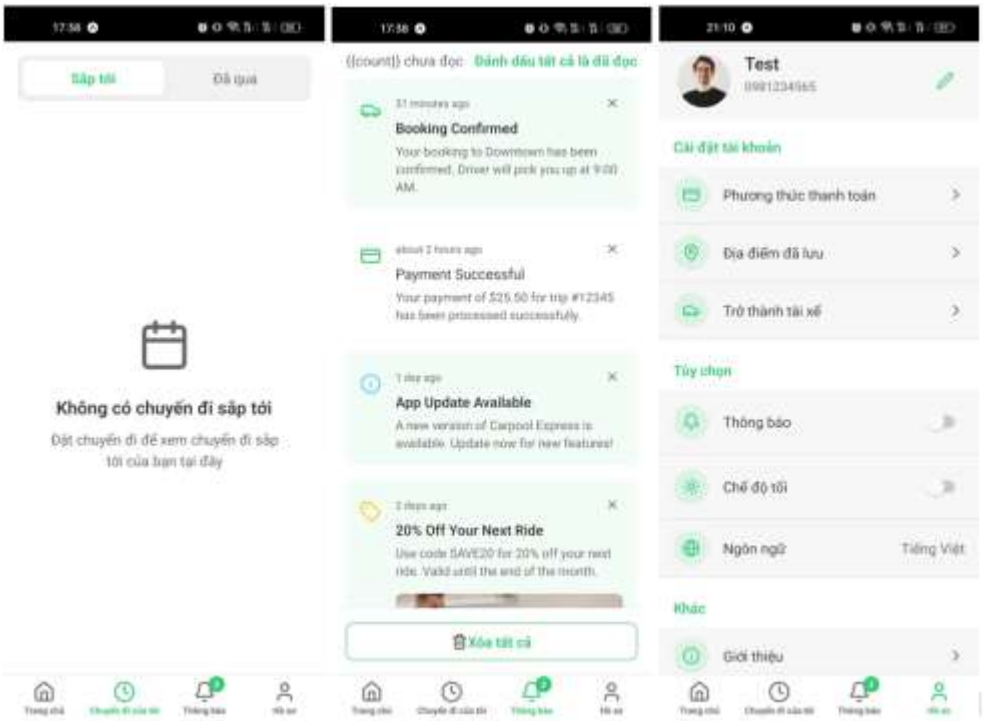
Hình 24. Màn cấu hình hệ thống trên website



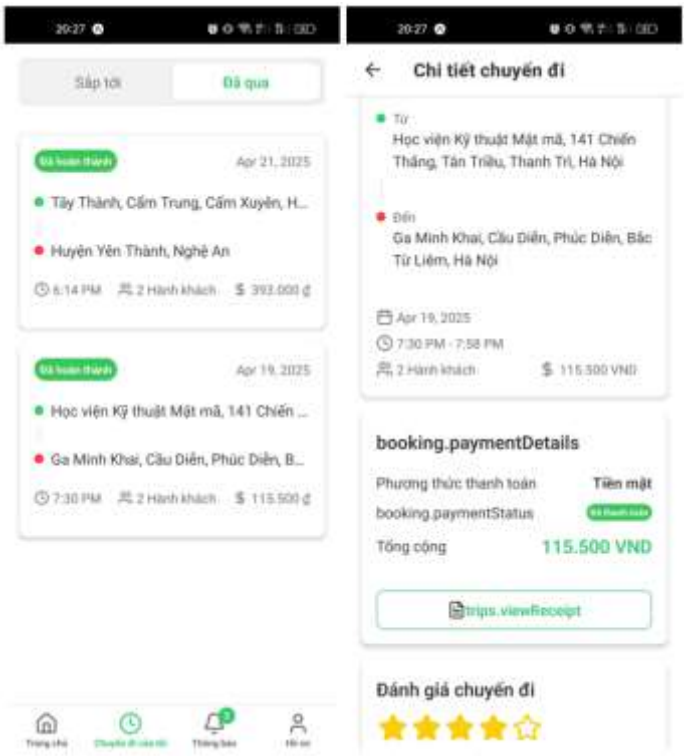
Hình 25. Màn quản lý tài khoản trên website



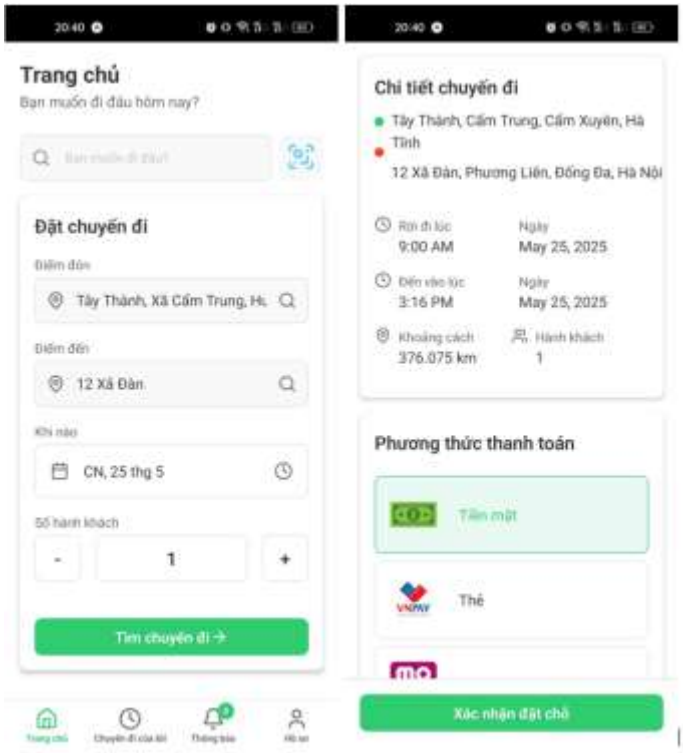
Hình 26. Màn hình đăng nhập



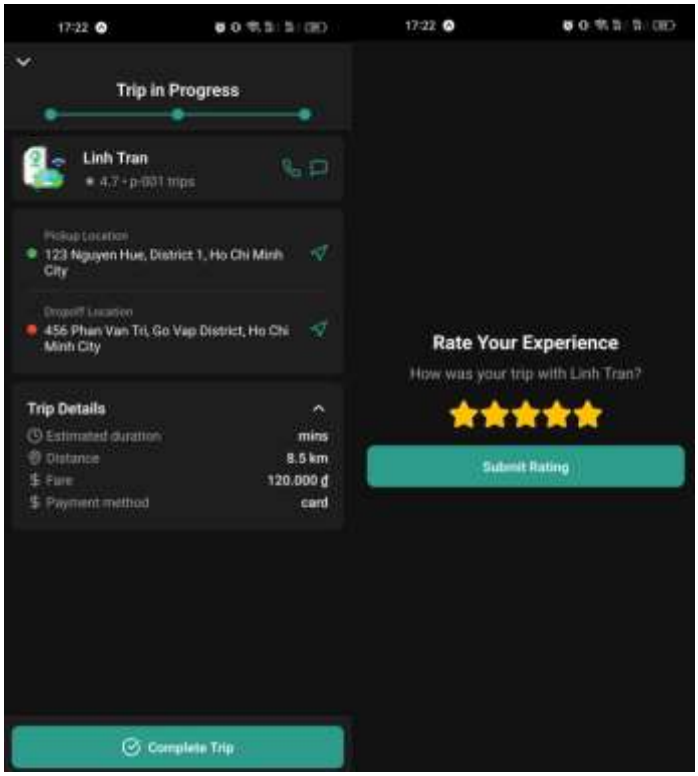
Hình 27. Giao diện chính của Ứng dụng Khách hàng



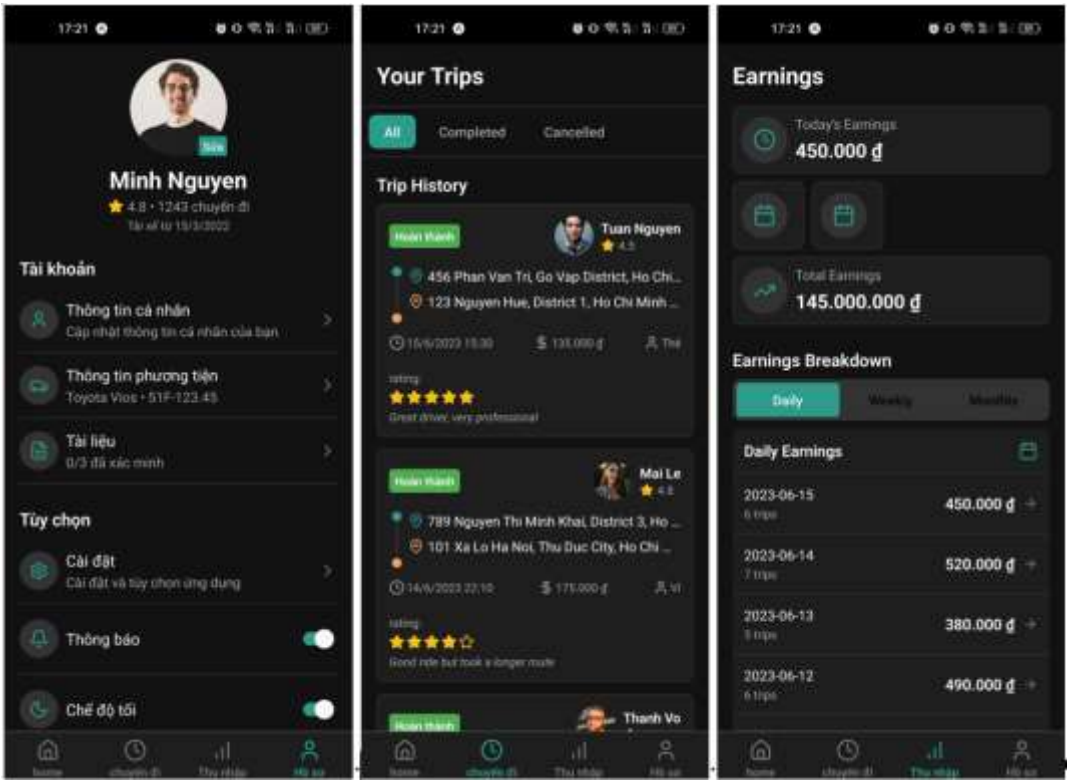
Hình 28. Màn hình chuyến đi của tôi



Hình 29. Màn hình đặt xe



Hình 30. Màn hình tiến trình chuyến xe



Hình 31. Giao diện chính của Ứng dụng Tài xế

3.5. Kiểm thử và đánh giá

3.5.1. Kế hoạch kiểm thử

- **Mục tiêu:** Đảm bảo hệ thống vận hành ổn định, đáp ứng yêu cầu chức năng và phi chức năng đã đề ra.
- **Phương pháp kiểm thử:** Thực hiện kiểm thử thủ công (Manual Testing) thông qua thao tác trực tiếp trên hệ thống web và ứng dụng di động. Các chức năng được kiểm tra dựa trên các kịch bản sử dụng thực tế.
- **Phạm vi kiểm thử:**
  - Kiểm thử chức năng đăng ký, đăng nhập và phân quyền người dùng.
  - Kiểm thử quy trình đặt xe và theo dõi chuyến đi cho cả người dùng và tài xế.
  - Kiểm thử quy trình đăng ký tài xế và xét duyệt từ phía admin.
  - Kiểm thử các chức năng quản lý trong giao diện website dành cho admin (tài khoản, tuyến xe, tài chính...).
- Công cụ hỗ trợ kiểm thử:
  - Postman: Kiểm thử các API thủ công (Manual API Testing).
  - Expo Go: Truy cập và kiểm tra ứng dụng di động trên thiết bị thật.

- Ghi chú / Bảng kiểm (Checklist): Theo dõi và ghi lại kết quả kiểm thử theo từng chức năng.

3.5.2. Thiết kế các ca kiểm thử

Bảng 24. Thiết kế các ca kiểm thử

ID	**Mô tả test case**	**Các bước thực hiện**	**Kết quả mong đợi**	**Kết quả thực tế**	**Trạng thái**
TC DN 01	Kiểm tra giao diện màn hình đăng nhập	Truy cập trang đăng nhập hoặc mở ứng dụng Rider/Driver	Hiển thị đúng giao diện đăng nhập	Hiển thị đúng giao diện đăng nhập	Pass
TC DN 02	Đăng nhập với thông tin hợp lệ	Nhập email và mật khẩu hợp lệ → nhấn nút "Đăng nhập"	Chuyển hướng đến trang chính của người dùng	Chuyển đúng trang chính	Pass
TC DN 03	Đăng nhập với thông tin không hợp lệ	Nhập sai email hoặc mật khẩu → nhấn "Đăng nhập"	Hiển thị thông báo lỗi "Thông tin đăng nhập không chính xác"	Hiển thị thông báo lỗi	Pass
TC DN 04	Kiểm tra phân quyền người dùng	Đăng nhập bằng tài khoản rider/driver/admin → truy cập các chức năng	Chỉ cho phép truy cập các chức năng phù hợp với vai trò	Phân quyền đúng	Pass
TC ACC 01	Kiểm tra hiển thị thông tin tài khoản	Đăng nhập và vào trang "Tài khoản của tôi"	Hiển thị đầy đủ thông tin cá nhân	Hiển thị đầy đủ	Pass
TC RD 01	Đặt chuyến đi mới (Rider)	Rider đăng nhập → chọn điểm đi/điểm đến → nhấn "Đặt xe"	Hệ thống tạo chuyến đi, chuyển sang trạng thái chờ tài xế	Hoạt động đúng quy trình	Pass

TC RD 02	Theo dõi chuyến đi đang diễn ra	Rider chọn chuyến đang hoạt động	Hiển thị bản đồ và thông tin tài xế, thời gian di chuyển	Hiển thị đúng	Pass
TC RD 03	Xem lịch sử các chuyến đi	Rider vào "Chuyến đi của tôi"	Hiển thị danh sách chuyến đã đặt và trạng thái hoàn tất	Hiển thị đúng	Pass
TC DV 01	Đăng ký tài xế	Người dùng chọn "Trở thành tài xế" → điền thông tin cá nhân, phương tiện, giấy tờ	Gửi yêu cầu thành công, chờ xét duyệt	Gửi yêu cầu thành công	Pass
TC AD 01	Admin duyệt yêu cầu tài xế	Admin đăng nhập → vào danh sách yêu cầu → chọn "Duyệt"	Trạng thái tài xế chuyển sang "Được duyệt", có thể hoạt động	Đã duyệt và cập nhật đúng trạng thái	Pass
TC AD 02	Admin quản lý tài khoản người dùng	Admin đăng nhập → vào "Quản lý tài khoản"	Hiển thị danh sách tài khoản, cho phép cập nhật trạng thái và chi tiết hồ sơ	Hiển thị và thao tác đúng	Pass
TC AD 03	Admin cấu hình hệ thống	Admin vào mục "Cấu hình" → thay đổi các cài đặt hệ thống (giá cước, tuyến...)	Cập nhật thành công và áp dụng thay đổi	Cập nhật đúng	Pass
TC SEC 01	Kiểm tra yêu cầu API khi chưa đăng nhập	Gửi yêu cầu đến API (Postman) mà không kèm token	Trả về lỗi 401 Unauthorized	Trả về đúng mã lỗi	Pass

TC SEC 02	Kiểm tra hệ thống từ chối truy cập trái vai trò	Đăng nhập bằng rider → truy cập API của admin	Trả về lỗi 403 Forbidden	Trả về đúng mã lỗi	Pass

3.5.3. Đánh giá kết quả kiểm thử

Quá trình kiểm thử được thực hiện theo phương pháp kiểm thử thủ công với các tình huống mô phỏng hành vi thực tế của người dùng trên cả ba nền tảng: Website quản trị (Admin Dashboard), ứng dụng người dùng (Rider App) và ứng dụng tài xế (Driver App). Các chức năng được kiểm tra bao gồm đăng nhập, phân quyền, đặt xe, quản lý tài khoản, đăng ký và xét duyệt tài xế, theo dõi chuyến đi và cấu hình hệ thống.

Kết quả kiểm thử cho thấy:

- Tất cả các chức năng chính hoạt động ổn định và đúng theo yêu cầu đặt ra.
- Phân quyền người dùng (rider, driver, admin) được áp dụng chính xác, đảm bảo tính bảo mật và giới hạn truy cập phù hợp.
- Các yêu cầu API được bảo vệ đúng cách, không cho phép truy cập trái phép hoặc sai vai trò.
- Giao diện người dùng hiển thị rõ ràng, các thao tác đặt xe, duyệt tài xế, theo dõi chuyến đi diễn ra mượt mà và phản hồi nhanh.
- Trong quá trình kiểm thử, một số lỗi nhỏ về giao diện (chưa đồng nhất font chữ, căn lề) đã được phát hiện và khắc phục kịp thời.

Nhìn chung, hệ thống đạt mức ổn định cao và sẵn sàng triển khai thực tế, đáp ứng đầy đủ yêu cầu chức năng và phi chức năng của đề tài.

Tổng kết chương 3

Chương 3 đã trình bày chi tiết quá trình xây dựng, triển khai và kiểm thử hệ thống. Qua đó, hệ thống đã hoàn thiện các chức năng cốt lõi và vận hành ổn định trên môi trường thực tế. Dù còn một số hạn chế nhỏ về giao diện và phạm vi thử nghiệm, nhưng kết quả kiểm thử cho thấy hệ thống đã sẵn sàng để áp dụng thử nghiệm mở rộng. Đây là cơ sở quan trọng để đề xuất các hướng phát triển tiếp theo trong chương Kết luận.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả

Trong quá trình thực hiện đồ án, các nhiệm vụ sau đã được hoàn thành:

- Phân tích và thiết kế hệ thống đặt xe ghép tuyến đa nền tảng, số hóa quy trình truyền thống.
- Xây dựng website và ứng dụng di động (khách hàng, tài xế) sử dụng ReactJS, React Native, Node.js, và MongoDB.
- Triển khai thử nghiệm với 50 khách hàng và 10 tài xế, đạt tỷ lệ đặt xe thành công 95% và thời gian phản hồi trung bình 200ms.
- Tiến hành kiểm tra hệ thống tổng thể nhằm đảm bảo tính ổn định và an toàn bảo mật.
- Đề xuất thuật toán gợi ý tuyến đường, giảm thời gian chờ, vượt trội hơn chọn tuyến cố định của Văn Minh.
- Tập trung vào dịch vụ xe ghép tuyến bán cố định, hỗ trợ các tuyến liên tỉnh – trong khi Grab/Be chủ yếu phục vụ nhu cầu cá nhân trong đô thị.

Hệ thống không chỉ cải thiện trải nghiệm người dùng mà còn tăng hiệu quả quản lý cho đơn vị vận tải, góp phần thúc đẩy chuyển đổi số trong ngành giao thông.

2. Khó khăn

- **Tích hợp GPS thời gian thực**: Độ chính xác định vị ở vùng nông thôn chưa cao, đòi hỏi cơ chế dự phòng (caching).
- **Tối ưu hiệu suất MongoDB**: Truy vấn danh sách tuyến đường ban đầu chậm do dữ liệu lớn, đã khắc phục bằng cách thêm index và phân trang (pagination).
- **Hạn chế thiết bị thử nghiệm**: Một số thiết bị Android cũ gặp lỗi giao diện, yêu cầu tối ưu hóa bổ sung.
- **Kiến thức bảo mật**: Việc áp dụng các tiêu chuẩn bảo mật đòi hỏi thời gian nghiên cứu chuyên sâu.

3. Hướng phát triển

- **Tích hợp trí tuệ nhân tạo**: Ứng dụng học máy để dự đoán nhu cầu đặt xe theo thời gian và tối ưu hóa tuyến ghép.
- **Mở rộng thanh toán**: Hỗ trợ ví điện tử (ZaloPay,...) và thẻ quốc tế để tăng tiện lợi.
- **Cải thiện hiệu suất vùng tín hiệu yếu**: Phát triển chế độ ngoại tuyến (offline mode) cho ứng dụng, cho phép lưu trữ tạm thời dữ liệu lộ trình.
- **Nâng cao bảo mật**: Tích hợp xác thực hai yếu tố (2FA) và kiểm tra định kỳ lỗ hổng bảo mật.
- **Mở rộng quy mô**: Triển khai hệ thống cho các loại hình vận tải khác như xe buýt hoặc xe du lịch liên tỉnh.

TÀI LIỆU THAM KHẢO

[1] Phạm Nguyễn Cương, Nguyễn Trần Minh Thư, Hồ Bảo Quốc**,** *Giáo trình phân tích thiết kế hệ thống thông tin theo hướng đối tượng*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội, 2016.

[2] W3Schools Node.js Tutorial, *Bài hướng dẫn chi tiết với các ví dụ thực hành từ W3Schools*. *https://www.w3schools.com/nodejs/default.asp*

[3] React Native Documentation. *Official React Native Guide*. https://reactnative.dev/docs/getting-started

[4] MongoDB Documentation. *MongoDB Manual*. https://www.mongodb.com/docs/manual/.

[5] Goong Map APIs, *Giới thiệu Tổng quan*. https://help.goong.io/kb/gioi-thieu-tong-quan/

[6] Vnpay, *Hướng dẫn tích hợp hệ thống PAY - Cổng thanh toán VNPAY*. https://sandbox.vnpayment.vn/apis/docs/thanh-toan-pay/pay.html