

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**ĐỒ ÁN TỐT NGHIỆP**

**Xây dựng ứng dụng đặt người lái xe hộ**

**Triệu Tuyên Nhâm**  
nham.tt204771@sis.hust.edu.vn

**Trường Công nghệ thông tin và truyền thông**

**Giảng viên hướng dẫn:** TS. Đỗ Tiên Dũng

Chữ ký GVHD

**Khoa:** Kỹ thuật máy tính

**Trường:** Công nghệ Thông tin và Truyền thông

**HÀ NỘI, 12/2024**

# LỜI CẢM ƠN

Trong quá trình thực hiện và hoàn thành Đồ án tốt nghiệp, em xin gửi lời cảm ơn sâu sắc đến tất cả những người đã đồng hành, động viên và hỗ trợ em trong suốt thời gian qua.

Trước tiên, em xin gửi lời cảm ơn chân thành tới TS. Đỗ Tiên Dũng, người thầy đã tận tâm chỉ dẫn và đưa ra những góp ý giúp đỡ em từng bước trong quá trình thực hiện đồ án.

Em cũng xin chân thành cảm ơn Đại học Bách Khoa Hà Nội, cùng quý thầy cô trong trường đặc biệt là các thầy cô của Trường Công nghệ Thông tin và Truyền thông, những người đã truyền đạt cho em những kiến thức quý giá và tạo điều kiện thuận lợi cho quá trình học tập và nghiên cứu của em.

Bên cạnh đó, em không thể không nhắc đến gia đình và những người bạn thân yêu. Chính sự quan tâm, động viên và chia sẻ của mọi người đã tiếp thêm sức mạnh để em vượt qua khó khăn và hoàn thành đồ án này.

Cuối cùng, em rất mong nhận được những góp ý từ thầy cô và các bạn để có thể cải thiện và hoàn thiện đồ án của mình hơn nữa.

Em xin chân thành cảm ơn!

# TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong những năm gần đây, số lượng phương tiện giao thông tại Việt Nam tăng nhanh chóng, đặc biệt là ở các thành phố lớn. Cùng với đó, nhu cầu sử dụng dịch vụ lái xe hộ ngày càng tăng cao trong nhiều tình huống như: người dùng mệt mỏi sau giờ làm việc, đã sử dụng đồ uống có cồn, không quen đường, hoặc không tự tin lái xe trong điều kiện giao thông phức tạp. Tuy nhiên, việc tìm kiếm một tài xế lái xe hộ đáng tin cậy vẫn còn gặp nhiều khó khăn.

Tại Việt Nam, trong khi dịch vụ xe ôm và taxi đã phổ biến, thì dịch vụ đặt người lái xe hộ vẫn còn khá mới mẻ và chưa được nhiều người biết đến. Xuất phát từ thực tế đó, đồ án này tập trung vào việc phát triển một ứng dụng di động cho phép người dùng dễ dàng đặt người lái xe hộ khi cần thiết. Mục tiêu của ứng dụng này là đem lại cho người dùng trải nghiệm tốt và dễ dàng để sử dụng.

Để có thể thực hiện được điều này, đồ án là sự kết hợp của framework Flutter, NestJS và Firebase. Bên cạnh đó còn có sự hỗ trợ đến từ API của Google Map Platform hỗ trợ việc tìm kiếm, định tuyến các địa điểm cũng như tính toán khoảng cách và độ ưu tiên trong việc phân phối tài xế 1 cách hiệu quả.

Qua đó, ứng dụng đã phần nào giải quyết được các vấn đề ban đầu, mang lại trải nghiệm mượt mà, tiện lợi cho người dùng. Ứng dụng không chỉ giải quyết được vấn đề tai nạn giao thông do sử dụng rượu bia mà còn góp phần nâng cao ý thức chấp hành luật an toàn giao thông của mọi người, từ đó tạo nên một môi trường giao thông văn minh, an toàn hơn.

Sinh viên thực hiện  
(Ký và ghi rõ họ tên)

## MỤC LỤC

|   |          |
|---|----------|
| <b>CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....</b>             | <b>1</b> |
| 1.1 Đặt vấn đề.....                                 | 1        |
| 1.2 Mục tiêu và phạm vi đề tài.....                 | 1        |
| 1.3 Định hướng giải pháp.....                       | 1        |
| 1.4 Bố cục đồ án .....                              | 2        |
| <b>CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU.....</b> | <b>3</b> |
| 2.1 Khảo sát hiện trạng .....                       | 3        |
| 2.2 Tổng quan chức năng .....                       | 4        |
| 2.2.1 Biểu đồ use case tổng quát .....              | 4        |
| 2.2.2 Biểu đồ use case phân rã Xác thực.....        | 5        |
| 2.2.3 Biểu đồ use case phân rã Xác thực .....       | 6        |
| 2.2.4 Biểu đồ use case phân rã Đặt tài xế .....     | 7        |
| 2.2.5 Biểu đồ use case phân rã Nhận chuyển.....     | 8        |
| 2.2.6 Quy trình nghiệp vụ Tìm kiếm tài xế.....      | 9        |
| 2.3 Đặc tả chức năng .....                          | 10       |
| 2.3.1 Đặc tả use case Đặt tài xế.....               | 10       |
| 2.3.2 Đặc tả use case Nhận chuyển.....              | 11       |
| 2.3.3 Đặc tả use case Tìm kiếm tài xế.....          | 12       |
| 2.3.4 Đặc tả use case Chính sửa thông tin .....     | 13       |
| 2.4 Yêu cầu phi chức năng .....                     | 13       |
| 2.4.1 Yêu cầu về hoạt động.....                     | 13       |
| 2.4.2 Yêu cầu về bảo mật .....                      | 13       |
| 2.4.3 Độ tin cậy, khả năng bảo trì.....             | 13       |
| 2.4.4 Trải nghiệm người dùng.....                   | 13       |

|   |           |
|---|-----------|
| <b>CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG.....</b>                         | <b>14</b> |
| 3.1 Flutter.....  | 14        |
| 3.1.1 Giới thiệu .....  | 14        |
| 3.1.2 Mục đích sử dụng.....                                     | 14        |
| 3.1.3 Ưu - Nhược điểm .....                                     | 14        |
| 3.2 NestJS .....  | 15        |
| 3.2.1 Giới thiệu .....  | 15        |
| 3.2.2 Mục đích sử dụng.....                                     | 15        |
| 3.2.3 Ưu - Nhược điểm .....                                     | 15        |
| 3.3 Firebase .....  | 16        |
| 3.3.1 Giới thiệu .....  | 16        |
| 3.3.2 Mục đích sử dụng.....                                     | 16        |
| 3.3.3 Ưu - Nhược điểm .....                                     | 16        |
| <b>CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG ....</b> | <b>17</b> |
| 4.1 Thiết kế kiến trúc.....                                     | 17        |
| 4.1.1 Lựa chọn kiến trúc phần mềm .....                         | 17        |
| 4.1.2 Thiết kế tổng quan.....                                   | 19        |
| 4.1.3 Thiết kế chi tiết gói .....                               | 20        |
| 4.2 Thiết kế chi tiết.....                                      | 21        |
| 4.2.1 Thiết kế giao diện .....                                  | 21        |
| 4.2.2 Thiết kế lớp .....  | 27        |
| 4.2.3 Thiết kế cơ sở dữ liệu .....                              | 30        |
| 4.3 Xây dựng ứng dụng.....                                      | 32        |
| 4.3.1 Thư viện và công cụ sử dụng.....                          | 32        |
| 4.3.2 Kết quả đạt được .....                                    | 32        |
| 4.3.3 Minh họa các chức năng chính .....                        | 33        |

|  |           |
|--|-----------|
| 4.4 Kiểm thử.....  | 43        |
| 4.4.1 Kiểm thử đăng nhập người dùng .....                                | 43        |
| 4.4.2 Kiểm thử tính chi phí chuyến đi.....                               | 43        |
| 4.5 Triển khai .....   | 44        |
| <b>CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT.....</b>                  | <b>45</b> |
| 5.1 Lưu trữ file.....  | 45        |
| 5.1.1 Vấn đề đặt ra.....   | 45        |
| 5.1.2 Giải pháp .....  | 45        |
| 5.1.3 Kết quả .....  | 46        |
| 5.2 Thuật toán tính toán độ ưu tiên của tài xế khi thực hiện đặt xe..... | 46        |
| 5.2.1 Vấn đề đặt ra.....   | 46        |
| 5.2.2 Giải pháp .....  | 46        |
| 5.2.3 Kết quả .....  | 47        |
| 5.3 Theo dõi lịch trình di chuyển của tài xế.....                        | 48        |
| 5.3.1 Vấn đề đặt ra.....   | 48        |
| 5.3.2 Giải pháp .....  | 48        |
| 5.3.3 Kết quả .....  | 50        |
| 5.4 Tính toán chi phí chuyến đi.....                                     | 50        |
| 5.4.1 Vấn đề đặt ra.....   | 50        |
| 5.4.2 Giải pháp .....  | 50        |
| 5.4.3 Kết quả .....  | 51        |
| <b>CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>                      | <b>53</b> |
| 6.1 Kết luận .....   | 53        |
| 6.2 Hướng phát triển.....  | 53        |



## DANH MỤC HÌNH VẼ

|           |   |    |
|-----------|---|----|
| Hình 2.1  | Biểu đồ use case tổng quát . . . . .                              | 4  |
| Hình 2.2  | Biểu đồ usecase Xác thực . . . . .                                | 5  |
| Hình 2.3  | Biểu đồ usecase Quản lý tài khoản . . . . .                       | 6  |
| Hình 2.4  | Biểu đồ usecase Đặt tài xế . . . . .                              | 7  |
| Hình 2.5  | Biểu đồ usecase Nhận chuyến . . . . .                             | 8  |
| Hình 2.6  | Quy trình nghiệp vụ Tìm kiếm tài xế . . . . .                     | 9  |
| <br>      |   |    |
| Hình 4.1  | Kiến trúc client-server . . . . .                                 | 17 |
| Hình 4.2  | Mô hình MVC . . . . .   | 18 |
| Hình 4.3  | Biểu đồ phụ thuộc gói Server . . . . .                            | 19 |
| Hình 4.4  | Biểu đồ phụ thuộc gói Driver . . . . .                            | 20 |
| Hình 4.5  | Ví dụ thiết kế gói . . . . .                                      | 21 |
| Hình 4.6  | Thiết kế chính các màn hình . . . . .                             | 22 |
| Hình 4.7  | Thiết kế màn hình Đặt tài xế . . . . .                            | 23 |
| Hình 4.8  | Thiết kế màn hình Thông tin người dùng . . . . .                  | 24 |
| Hình 4.9  | Thiết kế màn hình Lịch sử chuyến đi của người dùng . . . . .      | 25 |
| Hình 4.10 | Thiết kế màn hình Thông báo có chuyến đi mới của tài xế . . . . . | 26 |
| Hình 4.11 | Thiết kế lớp FirebaseService . . . . .                            | 27 |
| Hình 4.12 | Thiết kế lớp TravelService . . . . .                              | 27 |
| Hình 4.13 | Thiết kế lớp UserService . . . . .                                | 28 |
| Hình 4.14 | Thiết kế lớp DriverService . . . . .                              | 29 |
| Hình 4.15 | Biểu đồ thực thể liên kết . . . . .                               | 30 |
| Hình 4.16 | Màn hình thông báo chuyến đi của tài xế . . . . .                 | 34 |
| Hình 4.17 | Màn hình Tìm đặt tài xế của người dùng . . . . .                  | 36 |
| Hình 4.18 | Màn hình Lịch sử chuyến đi của người dùng . . . . .               | 38 |
| Hình 4.19 | Màn hình Tài khoản người dùng . . . . .                           | 40 |
| Hình 4.20 | Màn hình Lịch sử chuyến đi của tài xế . . . . .                   | 42 |
| <br>      |   |    |
| Hình 5.1  | Tải hình ảnh lên Firebase . . . . .                               | 45 |
| Hình 5.2  | Hiển thị URL hình ảnh trong database . . . . .                    | 46 |
| Hình 5.3  | Công thức Haversine . . . . .                                     | 47 |
| Hình 5.4  | Sắp xếp các tài xế theo độ ưu tiên . . . . .                      | 47 |
| Hình 5.5  | Cập nhật thay đổi vị trí của tài xế . . . . .                     | 49 |
| Hình 5.6  | Theo dõi vị trí di chuyển của tài xế . . . . .                    | 50 |
| Hình 5.7  | Tính toán chi phí chuyến đi . . . . .                             | 51 |

## **DANH MỤC BẢNG BIỂU**

|          |   |    |
|----------|---|----|
| Bảng 2.1 | Đặc tả usecase Đặt tài xế.                      | 10 |
| Bảng 2.2 | Đặc tả usecase Nhận chuyến.                     | 11 |
| Bảng 2.3 | Đặc tả usecase Tìm kiếm tài xế.                 | 12 |
| Bảng 2.4 | Đặc tả usecase Thay đổi thông tin cá nhân.      | 13 |
| <br>     |   |    |
| Bảng 4.1 | Giải thích về các thực thể . . . . .            | 30 |
| Bảng 4.2 | Thiết kế chi tiết thực thể tài xế . . . . .     | 31 |
| Bảng 4.3 | Thiết kế chi tiết thực thể chuyến đi . . . . .  | 31 |
| Bảng 4.4 | Thiết kế chi tiết thực thể người dùng . . . . . | 32 |
| Bảng 4.5 | Danh sách thư viện và công cụ sử dụng . . . . . | 32 |
| Bảng 4.6 | Thông kê thông tin ứng dụng . . . . .           | 33 |
| Bảng 4.7 | Kiểm thử đăng nhập người dùng . . . . .         | 43 |
| Bảng 4.8 | Kiểm thử tính chi phí chuyến đi . . . . .       | 44 |

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Trong những năm gần đây, tình trạng tai nạn giao thông liên quan đến việc sử dụng rượu bia đã trở thành một vấn đề nhức nhối không chỉ ở Việt Nam mà còn trên toàn thế giới. Việc sử dụng rượu bia làm suy giảm khả năng tập trung, giảm phản xạ và làm mất kiểm soát hành vi, dẫn đến nguy cơ cao xảy ra tai nạn giao thông nghiêm trọng. Vấn đề này đặt ra yêu cầu cấp thiết về việc nghiên cứu và triển khai các giải pháp nhằm giảm thiểu tai nạn giao thông liên quan đến rượu bia, hướng tới xây dựng một môi trường giao thông an toàn và văn minh.

Do đó em quyết định phát triển ứng dụng lái xe hộ - VISAFFE BK - trên nền tảng Android để giải quyết các vấn đề liên quan đến việc lái xe cho người dùng khi sử dụng các đồ uống có cồn. Hiện nay đã có một số ứng dụng giúp đặt tài xế lái hộ tuy nhiên các ứng dụng này còn một số nhược điểm như: giao diện phức tạp, không tiện lợi cho việc theo dõi di chuyển. Vì vậy ứng dụng này sẽ cung cấp tìm kiếm tài xế lái hộ cho người dùng một cách nhanh chóng, hiệu quả và tiện lợi.

## 1.2 Mục tiêu và phạm vi đề tài

Dựa trên những vấn đề đã nêu ở trên, mục tiêu của dự án này là xây dựng một ứng dụng giúp đơn giản hóa quá trình tìm kiếm và kết nối giữa người cần tài xế lái xe hộ và các tài xế sẵn sàng hỗ trợ. Mục tiêu cốt lõi của ứng dụng bao gồm:

- Cải thiện trải nghiệm người dùng: Tối ưu hóa giao diện và chức năng để người dùng dễ dàng đặt tài xế trong vài bước đơn giản.
- Nâng cao mức độ an toàn: Sử dụng công nghệ định vị GPS, theo dõi hành trình và hệ thống đánh giá tài xế để đảm bảo sự an tâm.
- Tạo giá trị cộng đồng: Góp phần giảm thiểu nguy cơ tai nạn giao thông trong các tình huống lái xe không an toàn (như say rượu, mệt mỏi).

## 1.3 Định hướng giải pháp

Để đạt được mục tiêu đề ra, ứng dụng sẽ được xây dựng dựa trên các giải pháp công nghệ hiện đại và chiến lược phát triển hợp lý. Ứng dụng sẽ được phát triển đa nền tảng bằng Flutter, đảm bảo hoạt động đồng nhất trên cả Android và iOS. Phần backend sử dụng NestJS với cấu trúc module hóa, tích hợp Firebase để cung cấp các tính năng thời gian thực như định vị, cập nhật trạng thái hành trình và lưu trữ dữ liệu người dùng. Hệ thống định vị Google Maps API sẽ được triển khai để hỗ trợ tìm kiếm địa điểm và theo dõi hành trình chính xác.

Về trải nghiệm người dùng, ứng dụng sẽ được thiết kế với giao diện đơn giản, thân thiện, giúp người dùng dễ dàng thao tác. Sau mỗi chuyến đi, người dùng có thể đánh giá tài xế để cải thiện chất lượng dịch vụ.

Để đảm bảo an toàn và minh bạch, ứng dụng sẽ yêu cầu tài xế và người dùng cung cấp thông tin cá nhân và giấy tờ liên quan để xác thực. Hành trình của người dùng sẽ được theo dõi theo thời gian thực. Thông tin tài xế và chuyến đi sẽ được lưu trữ trên Firebase, đảm bảo minh bạch và dễ dàng tra cứu khi cần.

#### 1.4 Bố cục đồ án

Phần còn lại của báo cáo Đồ án tốt nghiệp này được tổ chức như sau:

- **Chương 2:** Dựa trên những vấn đề đã được nêu ở Chương 1, chương này sẽ phân tích chi tiết các chức năng và yêu cầu mà ứng dụng cần có. Kết thúc chương, các tiêu chí chức năng cụ thể mà ứng dụng cần đáp ứng sẽ được phác thảo dựa trên phân tích này.
- **Chương 3:** Trình bày chi tiết về các công nghệ được sử dụng trong ứng dụng cùng với mục đích và những ưu - nhược điểm của chúng.
- **Chương 4:** Dựa trên các yêu cầu chức năng đã thiết lập từ Chương 2, chương này trình bày chi tiết thiết kế ứng dụng từ kiến trúc tổng thể đến các thành phần hệ thống cụ thể. Phần đầu chương sẽ giới thiệu kiến trúc tổng thể, phác thảo các thành phần và sự tương tác giữa chúng. Tiếp theo, chương đi sâu vào thiết kế chi tiết của từng thành phần, cung cấp sơ đồ minh họa và thông số kỹ thuật chức năng. Ngoài ra, chương cũng bao gồm thiết kế giao diện người dùng và cấu trúc dữ liệu của ứng dụng.
- **Chương 5:** Chương này nêu bật những đóng góp cá nhân trong quá trình phát triển ứng dụng, trình bày các chức năng hoặc phương pháp cụ thể mà em đã trực tiếp triển khai.
- **Chương 6:** Chương cuối cùng tóm tắt các khía cạnh đã hoàn thành của luận văn, bao gồm chức năng đã triển khai, công nghệ được sử dụng và thiết kế tổng thể của ứng dụng. Cuối chương sẽ thảo luận các hướng phát triển tiềm năng trong tương lai, phác thảo các tính năng hoặc chức năng có thể được bổ sung để nâng cao chất lượng ứng dụng.

## CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

### 2.1 Khảo sát hiện trạng

Hiện nay trên thị trường đã có một số ứng dụng hỗ trợ việc tìm và đặt người lái hộ như: FastGo, ViSafe, GOCheap,... Dưới đây là đánh giá về các ứng dụng này để tìm hiểu về các tính năng cũng như hạn chế của chúng:

**FastGo:** FastGo là một trong những ứng dụng thuê lái xe hộ khi say được yêu thích hiện nay. Được phát triển bởi công ty cổ phần FastGo Việt Nam, app này cung cấp dịch vụ lái xe chuyên nghiệp và an toàn cho người sử dụng.

- Ưu điểm:
  - Giao diện đơn giản, dễ sử dụng
  - Bảo mật thông tin cá nhân
  - Tính năng đặt xe và theo dõi hành trình dễ dàng
  - Hỗ trợ khách hàng 24/7

- Hạn chế:
  - Cần phải kết nối mạng ổn định
  - Không có tính năng đón khách

**ViSafe:** ViSafe là một ứng dụng thuê lái xe hộ khi say được phát triển bởi Công ty Cổ Phần An Toàn Giao Thông Việt Nam. App này nhắm đến việc cung cấp dịch vụ an toàn và chất lượng cho người sử dụng.

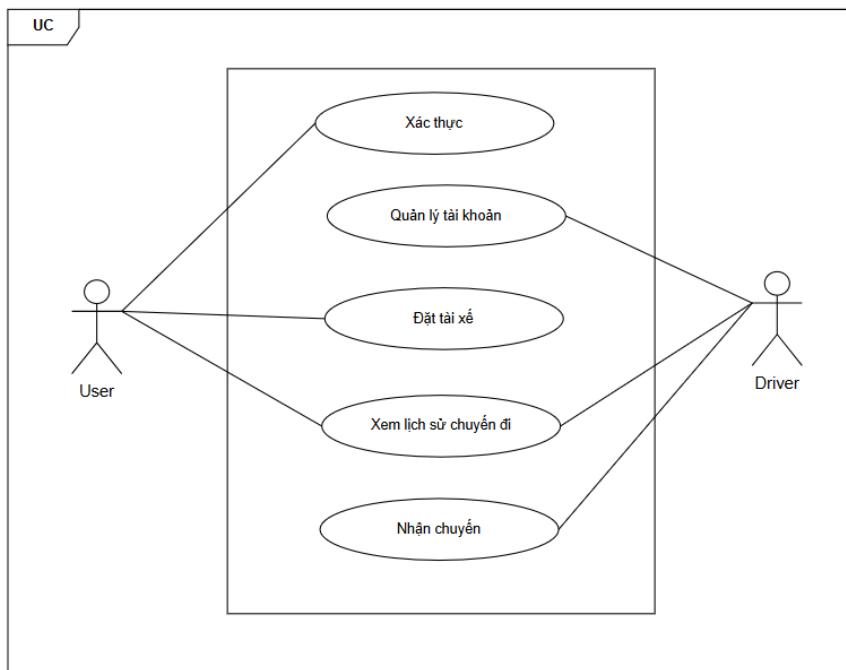
- Ưu điểm:
  - Giao diện đơn giản, dễ sử dụng
  - Bảo mật thông tin cá nhân
  - Hỗ trợ khách hàng 24/7
  - Đa dạng tính năng
- Hạn chế:
  - Không có tính năng theo dõi hành trình

**GOCheap:** GOCheap là một trong những app đặt lái xe hộ khi say tuyệt vời nhất ở thời điểm hiện tại. Ứng dụng này được phát triển và quản lý bởi Công ty TNHH GOCheap. Ứng dụng này nhắm đến mục tiêu cung cấp dịch vụ thuê lái xe hộ chất lượng và uy tín nhất cho người dùng.

- Ưu điểm:
  - Tiện lợi, nhanh chóng
  - Dịch vụ chất lượng
  - Bảo mật tốt
  - Hỗ trợ 24/7
- Hạn chế:
  - Giao diện chưa không thân thiện

## 2.2 Tổng quan chức năng

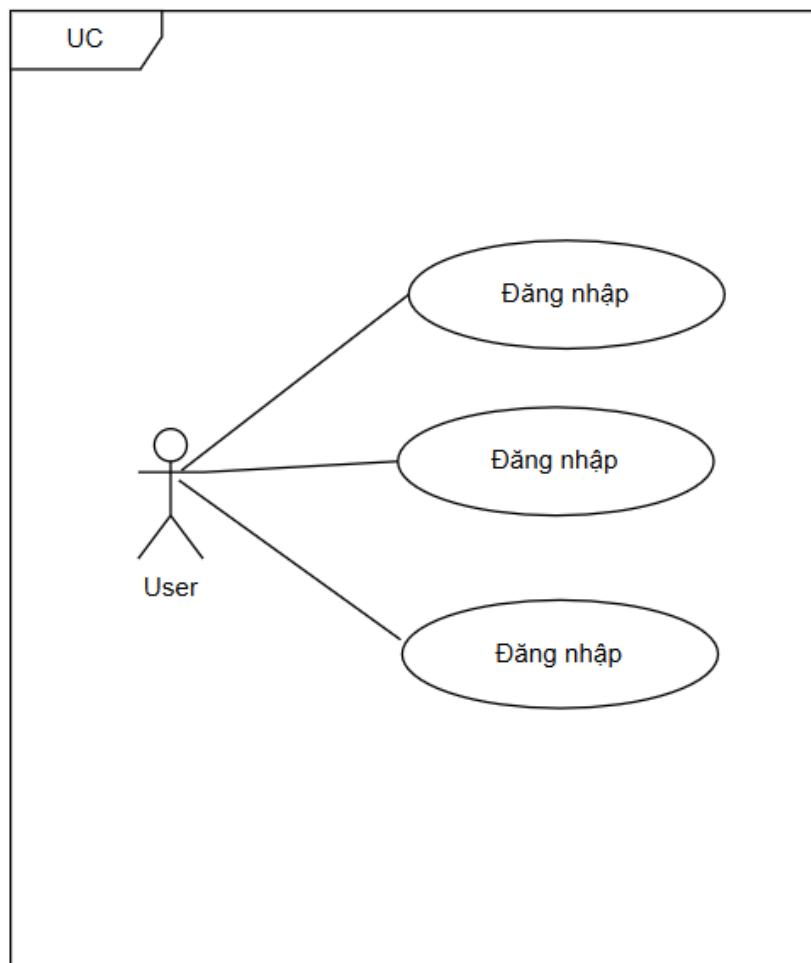
### 2.2.1 Biểu đồ use case tổng quát



**Hình 2.1:** Biểu đồ use case tổng quát

Hệ thống gồm 2 tác nhân là User và Driver. Để sử dụng các chức năng trong ứng dụng thì cả user và driver đều cần đăng nhập bằng số điện thoại và mật khẩu. Hệ thống bao gồm các usecase chính sau (i) usecase xác thực: xử lý việc đăng nhập, đăng ký của người dùng; (ii) usecase quản lý tài khoản: quản lý thông tin cá nhân, phương tiện của người dùng và tài xế; (iii) usecase đặt tài xế: tìm kiếm địa điểm, xem giá cước, chọn phương tiện, đặt tài xế; (iv) usecase xem lịch sử: người dùng xem lại lịch sử các chuyến đi của mình; (v) usecase nhận chuyến: tài xế nhận được thông tin về chuyến đi, xác nhận hoặc hủy chuyến

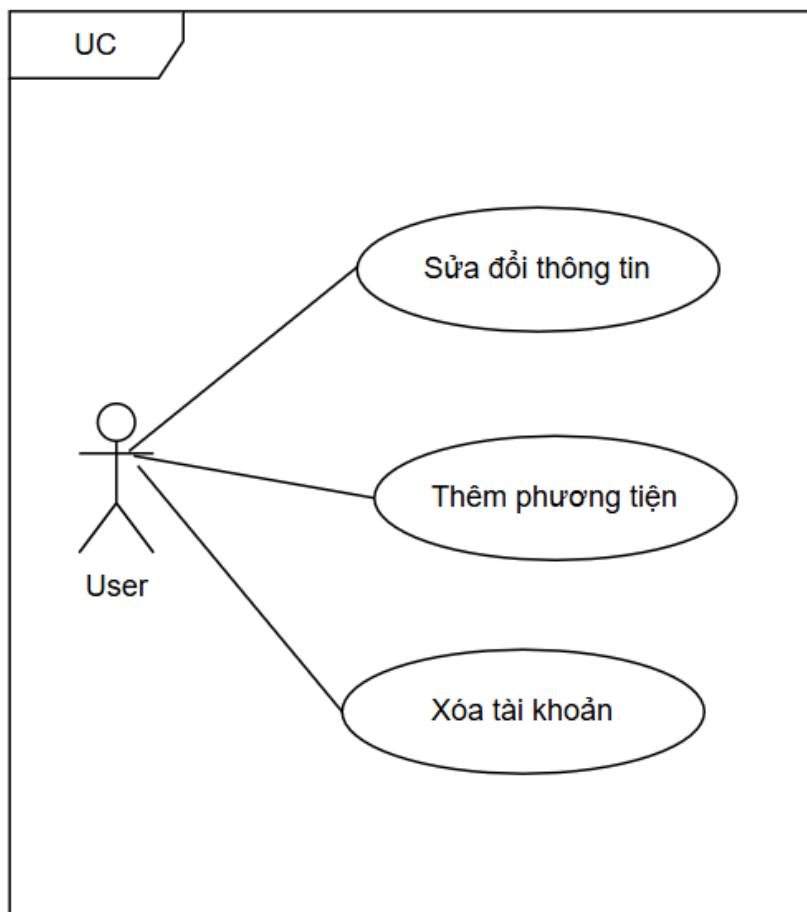
### 2.2.2 Biểu đồ use case phân rã Xác thực



**Hình 2.2:** Biểu đồ usecase Xác thực

Use case Xác thực bao gồm những chức năng chính sau: (i) đăng nhập: người dùng nhập số điện thoại và mật khẩu để đăng nhập vào hệ thống, (ii) đăng ký: người dùng nhập các thông tin bắt buộc để đăng ký, (iii) đăng xuất: Đăng xuất khỏi tài khoản đang được đăng nhập.

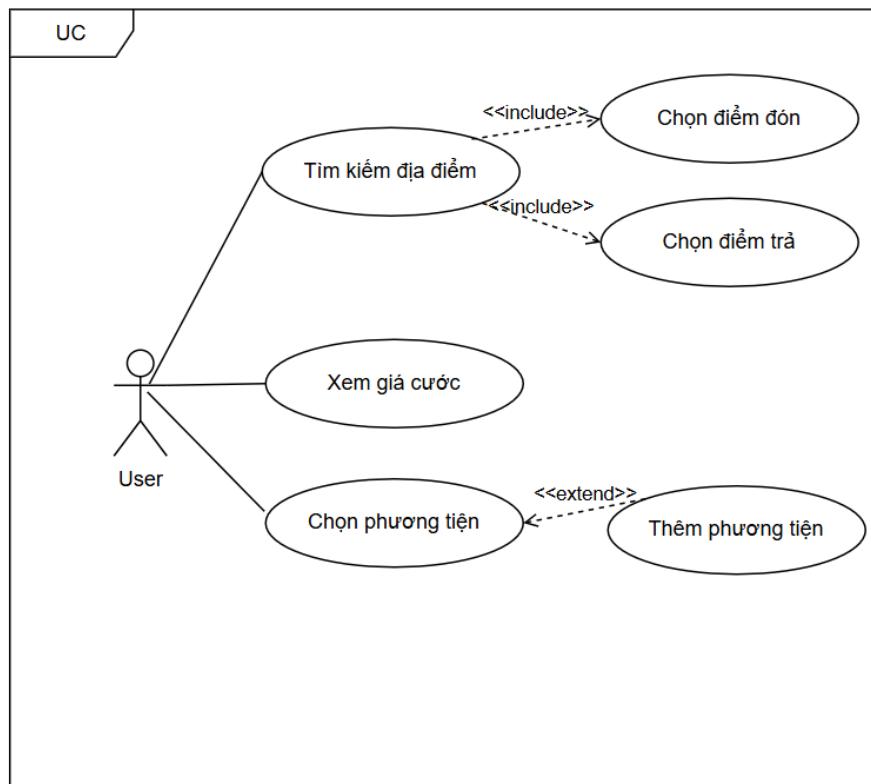
### 2.2.3 Biểu đồ use case phân rã Xác thực



**Hình 2.3:** Biểu đồ usecase Quản lý tài khoản

Usecase Quản lý tài khoản bao gồm những chức năng chính sau (i) sửa đổi thông tin cá nhân: người dùng sửa đổi các thông tin cá nhân của mình, (ii) thêm phương tiện: người dùng thêm phương tiện muốn lái hộ, (iii) xóa tài khoản: người dùng xóa tài khoản của mình.

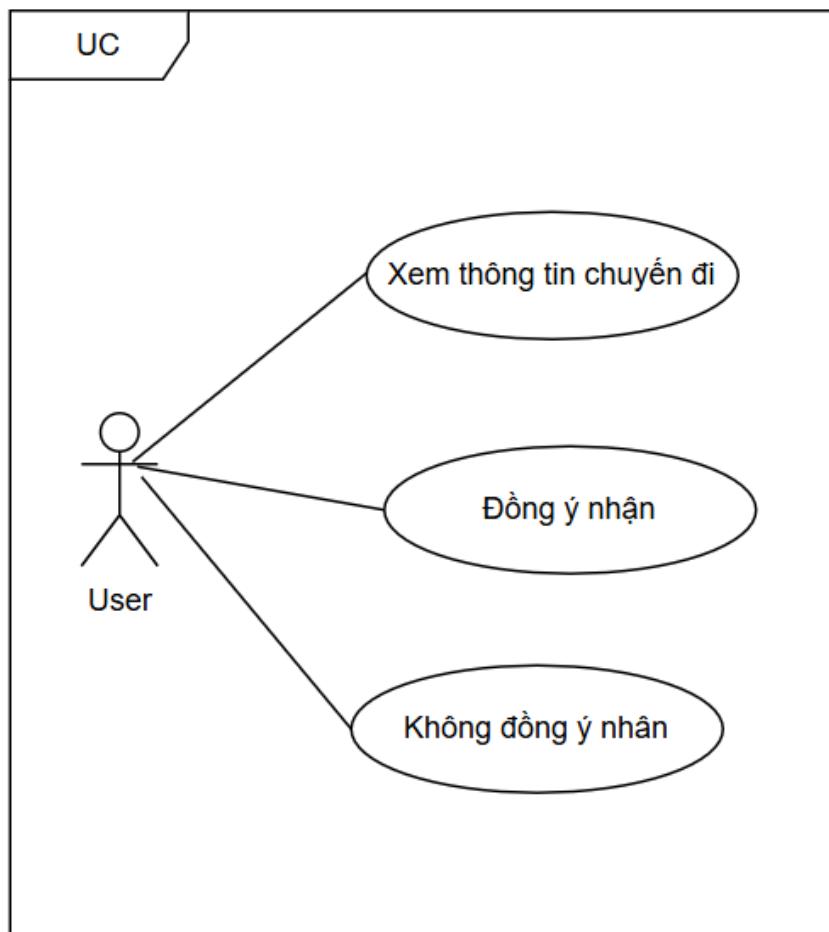
### 2.2.4 Biểu đồ use case phân rã Đặt tài xế



**Hình 2.4:** Biểu đồ usecase Đặt tài xế

Use case Đặt tài xế bao gồm những chức năng chính sau (i) tìm kiếm địa điểm: người dùng nhập điểm đến, điểm đón, (ii) xem giá cước: người dùng xem giá cước của các chuyến đi, (iii) chọn phương tiện: người dùng chọn phương tiện để đặt chuyến.

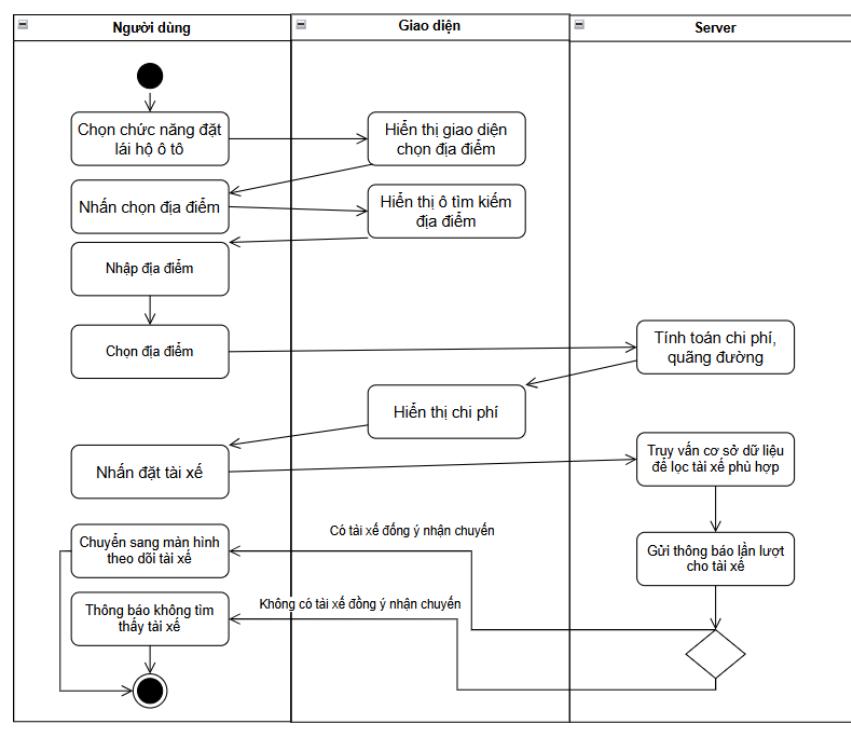
### 2.2.5 Biểu đồ use case phân rã Nhận chuyến



**Hình 2.5:** Biểu đồ usecase Nhận chuyến

Use case Nhận chuyến bao gồm những chức năng chính sau (i) xem thông tin về chuyến đi: tài xế xem thông tin về chuyến đi như giá cước, địa điểm, (ii) đồng ý nhận: tài xế chấp nhận chuyến xe này, (iii) không đồng ý nhận: tài xế không chấp nhận chuyến đi và bỏ qua chuyến đi.

### 2.2.6 Quy trình nghiệp vụ Tìm kiếm tài xế



**Hình 2.6:** Quy trình nghiệp vụ Tìm kiếm tài xế

Quy trình ở hình 2.6 mô tả nghiệp vụ tìm kiếm tài xế. Người dùng có nhu cầu đặt lái hộ sẽ nhấn vào nút "Ô tô" trong phần "Dịch vụ lái xe hộ" ở màn hình chính. Hệ thống sẽ hiển thị giao diện chọn vị trí để người dùng chọn và nhập vị trí. Khi nhập vị trí, hệ thống sẽ hiển thị gợi ý vị trí mà người dùng muốn đến. Sau khi người dùng chọn vị trí xong thì sẽ nhấn vào nút "Tiếp tục" để gửi dữ liệu tới server để thực hiện tính toán chi phí, quãng đường và trả về cho người dùng. Người dùng nhấn nút "Đặt tài xế", server sẽ truy vấn cơ sở dữ liệu tìm những tài xế phù hợp và gửi thông báo lần lượt cho các tài xế và chờ phản hồi từ tài xế đó. Nếu không có tài xế nào đồng ý thì thông báo không tìm được tài xế tới người dùng. Nếu có tài xế đồng ý thì chuyển sang màn hình chuyển di để theo dõi tài xế di chuyển tới vị trí đón.

### 2.3 Đặc tả chức năng

#### 2.3.1 Đặc tả use case Đặt tài xế

|                               |  |
|-------------------------------|--|
| <b>Mã usecase</b>             | UC-1   |
| <b>Tên usecase</b>            | Đặt tài xế   |
| <b>Mô tả</b>                  | Use case này mô tả quá trình đặt tài xế của người dùng   |
| <b>Tác nhân</b>               | Người dùng   |
| <b>Tiền điều kiện</b>         | Người dùng đăng nhập vào hệ thống  |
| <b>Luồng sự kiện chính</b>    | <ol style="list-style-type: none"> <li>1. Người dùng từ màn hình chính nhấn vào nút "Ô tô" của phần đặt xe hộ</li> <li>2. Người dùng tìm kiếm địa điểm đón và địa điểm trả</li> <li>3. Người dùng nhấn nút "Tiếp tục"</li> <li>4. Server tính toán quãng đường và giá tiền</li> <li>5. Người dùng xem quãng đường đi chuyển và giá tiền</li> <li>6. Người chọn phương tiện cần lái hộ</li> <li>7. Người dùng nhấn nút "Đặt tài xế"</li> <li>8. Server tìm kiếm tài xế phù hợp</li> </ol> |
| <b>Luồng sự kiện thay thế</b> | 8a. Hệ thống thông báo cho người dùng biết không tìm được tài xế   |
| <b>Hậu điều kiện</b>          | Hệ thống chuyển sang màn hình theo dõi tài xế di chuyển  |
| <b>Luồng ngoại lệ</b>         | Không  |

Bảng 2.1: Đặc tả usecase Đặt tài xế.

### 2.3.2 Đặc tả use case Nhận chuyến

|                               |  |
|-------------------------------|--|
| <b>Mã usecase</b>             | UC-2   |
| <b>Tên usecase</b>            | Nhận chuyến  |
| <b>Mô tả</b>                  | Use case này mô tả quá trình nhận chuyến của tài xế  |
| <b>Tác nhân</b>               | Tài xế   |
| <b>Tiền điều kiện</b>         | <ul style="list-style-type: none"> <li>- Người dùng đăng nhập vào hệ thống</li> <li>- Người dùng ở chế độ online</li> </ul>  |
| <b>Luồng sự kiện chính</b>    | <ol style="list-style-type: none"> <li>1. Hệ thống gửi thông báo tới tài xế đang có người muốn đặt chuyến</li> <li>2. Tài xế nhận thông báo về thông tin chuyến đi</li> <li>3. Người dùng nhấn nút "Tiếp tục"</li> </ol> |
| <b>Luồng sự kiện thay thế</b> | <ol style="list-style-type: none"> <li>3a. Người dùng ấn nút 'X' để không chấp nhận chuyến xe</li> <li>3b. Hệ thống chờ 10 giây nếu người dùng không thực hiện thao tác thì tự động bỏ qua chuyến xe.</li> </ol>         |
| <b>Hậu điều kiện</b>          | Hệ thống chuyển sang màn hình chuyến đi  |
| <b>Luồng ngoại lệ</b>         | Không  |

**Bảng 2.2:** Đặc tả usecase Nhận chuyến.

### 2.3.3 Đặc tả use case Tìm kiếm tài xế

|                               |   |
|-------------------------------|---|
| <b>Mã usecase</b>             | UC-3  |
| <b>Tên usecase</b>            | Tìm kiếm tài xế   |
| <b>Mô tả</b>                  | Use case này mô tả quá trình tìm kiếm tài xế phù hợp  |
| <b>Tác nhân</b>               | Người dùng, Hệ thống, Tài xế  |
| <b>Tiền điều kiện</b>         | <ul style="list-style-type: none"> <li>- Người dùng đăng nhập vào hệ thống</li> <li>- Tài xế ở chế độ online</li> </ul>   |
| <b>Luồng sự kiện chính</b>    | <ol style="list-style-type: none"> <li>1. Người dùng gửi thông tin về chuyến đi đến server</li> <li>2. Server lọc ra những người phù hợp</li> <li>3. Server sắp xếp danh sách những người phù hợp</li> <li>4. Server gửi thông báo lần lượt tới danh sách tài xế</li> </ol> |
| <b>Luồng sự kiện thay thế</b> | Không   |
| <b>Hậu điều kiện</b>          | Server gửi thông báo cho người dùng đã tìm được tài xế  |
| <b>Luồng ngoại lệ</b>         | Hệ thống gửi thông báo không tìm được tài xế  |

**Bảng 2.3:** Đặc tả usecase Tìm kiếm tài xế.

### 2.3.4 Đặc tả use case Chính sửa thông tin

|                               |   |
|-------------------------------|---|
| <b>Mã usecase</b>             | UC-4  |
| <b>Tên usecase</b>            | Thay đổi thông tin cá nhân  |
| <b>Mô tả</b>                  | Use case này mô tả quá trình thay đổi thông tin cá nhân   |
| <b>Tác nhân</b>               | Người dùng  |
| <b>Tiền điều kiện</b>         | - Người dùng đăng nhập vào hệ thống   |
| <b>Luồng sự kiện chính</b>    | <ol style="list-style-type: none"> <li>1. Người dùng nhấn nút "Tài khoản"</li> <li>2. Hệ thống hiển thị thông tin của người dùng</li> <li>3. Người dùng nhập thông tin cần thay đổi</li> <li>4. Người dùng nhấn nút "Cập nhật"</li> </ol> |
| <b>Luồng sự kiện thay thế</b> | 4a. Hệ thống thông báo cập nhật thành   |
| <b>Hậu điều kiện</b>          | Hệ thống hiển thị thông tin mới được chỉnh sửa  |
| <b>Luồng ngoại lệ</b>         | Hệ thống gửi thông báo chỉnh sửa thông tin thất bại   |

**Bảng 2.4:** Đặc tả usecase Thay đổi thông tin cá nhân.

## 2.4 Yêu cầu phi chức năng

Trong phần này, sinh viên đưa ra các yêu cầu khác nếu có, bao gồm các yêu cầu phi chức năng như hiệu năng, độ tin cậy, tính dễ dùng, tính dễ bảo trì, hoặc các yêu cầu về mặt kỹ thuật như về CSDL, công nghệ sử dụng, v.v.

### 2.4.1 Yêu cầu về hoạt động

Hệ thống hoạt động tốt trên các thiết bị Android 10 trở lên. Hệ thống có thể được truy cập từ bất kỳ đâu trên lãnh thổ Việt Nam

### 2.4.2 Yêu cầu về bảo mật

Người dùng bắt buộc phải đăng nhập để có thể sử dụng hệ thống

### 2.4.3 Độ tin cậy, khả năng bảo trì

Hệ thống không được tạo ra lỗi nghiêm trọng. Bất kỳ lỗi nào phải được khắc phục trong vòng 2 ngày

### 2.4.4 Trải nghiệm người dùng

Hệ thống có giao diện trực quan và dễ sử dụng. Dữ liệu được hiển thị một cách rõ ràng, dễ hiểu

## CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

### 3.1 Flutter

#### 3.1.1 Giới thiệu

Flutter là một framework phát triển giao diện người dùng (UI) mã nguồn mở do Google phát triển. Nó cho phép xây dựng các ứng dụng đa nền tảng (cross-platform) từ một cơ sở mã duy nhất. Flutter sử dụng ngôn ngữ lập trình Dart và cung cấp một bộ widget phong phú để tạo giao diện người dùng đẹp mắt và linh hoạt.

Flutter được sử dụng để phát triển ứng dụng di động (Android, iOS), ứng dụng web, và thậm chí cả ứng dụng desktop (Windows, MacOS, Linux).

#### 3.1.2 Mục đích sử dụng

Trong dự án này Flutter được sử dụng để xây dựng UI cho các màn hình của ứng dụng.

#### 3.1.3 Ưu - Nhược điểm

##### 1. Ưu điểm

- Phát triển đa nền tảng: Với mã nguồn duy nhất, Flutter cho phép phát triển ứng dụng cho nhiều nền tảng, giảm thời gian và công sức cho việc phát triển và bảo trì.
- Hiệu suất cao: Flutter biên dịch mã trực tiếp sang mã máy (native) thông qua Dart's Ahead-of-Time (AOT) compilation, đảm bảo hiệu suất gần như ứng dụng gốc (native).
- Giao diện đẹp và linh hoạt: Flutter cung cấp một bộ widget phong phú, được tùy chỉnh cao, giúp tạo ra giao diện người dùng đẹp mắt và đồng nhất trên các nền tảng.
- Cộng đồng phát triển mạnh mẽ: Flutter có một cộng đồng rộng lớn và tài liệu phong phú, giúp hỗ trợ giải quyết vấn đề nhanh chóng.

##### 2. Nhược điểm

- Dung lượng ứng dụng lớn: Các ứng dụng Flutter thường có dung lượng lớn hơn so với ứng dụng native, đặc biệt khi so với ứng dụng iOS.
- Thiếu một số thư viện native: Một số tính năng hoặc thư viện native cần phải được phát triển thêm bằng cách viết mã native (Android: Kotlin/Java, iOS: Swift/Objective-C).

- Hỗ trợ native không đầy đủ: Mặc dù Flutter cung cấp rất nhiều tính năng, nhưng với các ứng dụng cần tích hợp sâu vào hệ điều hành (như sử dụng Bluetooth, cảm biến), sẽ phải viết mã native thủ công.

### 3.2 NestJS

#### 3.2.1 Giới thiệu

NestJS là một framework mạnh mẽ để xây dựng ứng dụng phía server (backend) trong Node.js. Nó được xây dựng trên nền tảng của TypeScript và sử dụng các concept phổ biến từ Angular như Dependency Injection, Decorators, và Modularity. NestJS kết hợp các đặc điểm tốt nhất của các framework hiện có (như Express hoặc Fastify) và cung cấp một cấu trúc ứng dụng rõ ràng, dễ mở rộng.

#### 3.2.2 Mục đích sử dụng

- NestJS được sử dụng để viết các API cho ứng dụng
- Kết hợp với Firebase, API của Google Map Platform để xây dựng các dịch vụ backend

#### 3.2.3 Ưu - Nhược điểm

##### 1. Ưu điểm

- Kiến trúc module rõ ràng, dễ mở rộng: NestJS cung cấp cấu trúc module có tổ chức, giúp phát triển và bảo trì code dễ dàng hơn.
- Hỗ trợ TypeScript: NestJS được xây dựng trên TypeScript, giúp phát triển ứng dụng một cách an toàn và hiệu quả hơn.
- Dependency Injection: NestJS sử dụng Dependency Injection để quản lý các dependencies giữa các module và services, giúp code trở nên linh hoạt và dễ dàng trong việc kiểm thử phần mềm.
- Cộng đồng và tài liệu phong phú: NestJS có một cộng đồng đông đảo và tài liệu chính thức chi tiết giúp lập trình viên học tập và triển khai 1 cách dễ dàng.

##### 2. Nhược điểm

- Khó tiếp cận với người mới bắt đầu: Cấu trúc module, dependency injection và các khái niệm khác của NestJS có thể gây khó khăn cho người mới bắt đầu.
- Kích thước bundle: Do sử dụng TypeScript và nhiều tính năng, kích thước bundle của ứng dụng NestJS có thể lớn hơn so với các ứng dụng sử dụng các framework nhẹ hơn.

### 3.3 Firebase

#### 3.3.1 Giới thiệu

Firebase là một nền tảng phát triển ứng dụng di động và web do Google cung cấp. Nó cung cấp nhiều dịch vụ đám mây (Backend-as-a-Service) giúp các nhà phát triển tập trung vào việc xây dựng ứng dụng mà không cần quản lý cơ sở hạ tầng phía sau. Firebase cung cấp nhiều tính năng như xác thực người dùng, cơ sở dữ liệu thời gian thực, lưu trữ đám mây, phân tích và nhiều dịch vụ khác.

#### 3.3.2 Mục đích sử dụng

Trong dự án này, Firebase được sử dụng để:

- Realtime Database: Lưu trữ và quản lý dữ liệu thời gian thực
- Cloud Storage: Lưu trữ và quản lý files và hình ảnh
- Cloud Messaging: Gửi thông báo đẩy (push notifications) đến người dùng

#### 3.3.3 Ưu - Nhược điểm

##### 1. Ưu điểm

- Phát triển nhanh chóng: Firebase cung cấp nhiều dịch vụ được xây dựng sẵn, giúp giảm thời gian phát triển ứng dụng.
- Khả năng mở rộng: Firebase được xây dựng trên cơ sở hạ tầng của Google, cho phép ứng dụng dễ dàng mở rộng quy mô.
- Bảo mật cao: Được hỗ trợ bởi Google với các tính năng bảo mật mạnh mẽ.
- Hỗ trợ đa nền tảng: Firebase hỗ trợ phát triển ứng dụng cho nhiều nền tảng như iOS, Android và web.

##### 2. Nhược điểm

- Giá thành cao: Firebase có giá thành cao, đặc biệt là với các dịch vụ nâng cao.
- Phụ thuộc vào Google: Hoàn toàn phụ thuộc vào hạ tầng của Google.
- Tính linh hoạt hạn chế: Không thể tùy chỉnh sâu như các giải pháp backend tự xây dựng.

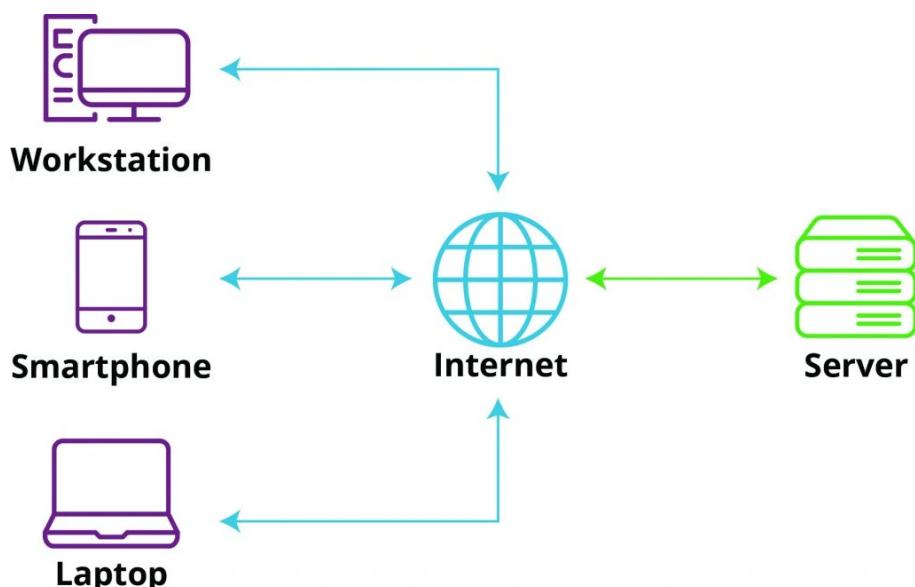
## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

### 4.1 Thiết kế kiến trúc

#### 4.1.1 Lựa chọn kiến trúc phần mềm

ĐATN này được xây dựng trên kiến trúc client-server. Đây là một kiến trúc được sử dụng rộng rãi cho các ứng dụng web, app mobile.

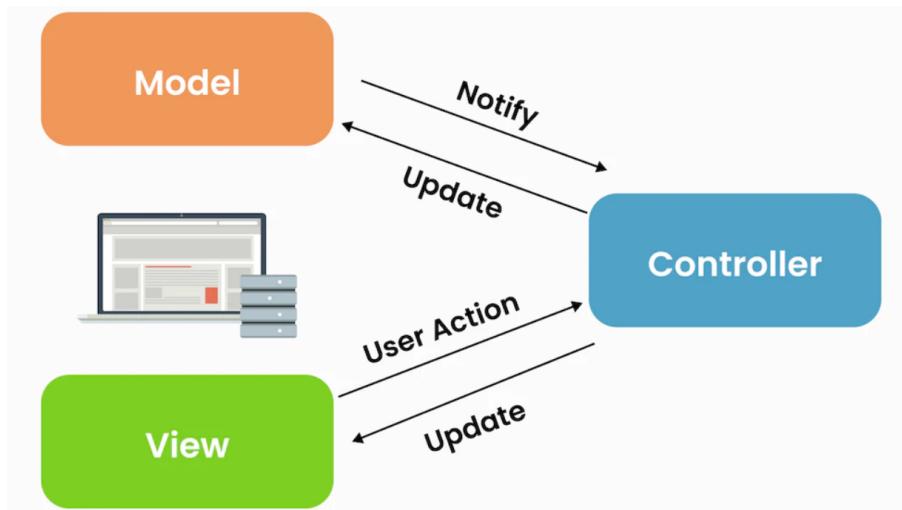
**Client:** Máy khách gửi yêu cầu tới máy chủ, nhận phản hồi và hiển thị dữ liệu nhận được ra cho người dùng. **Server:** Máy chủ nhận yêu cầu từ client, xử lý và trả về dữ liệu cho máy khách. **Database:** Lưu trữ dữ liệu và cung cấp dữ liệu cho máy chủ.



Hình 4.1: Kiến trúc client-server

MVC (Model-View-Controller) là một mô hình kiến trúc phần mềm được sử dụng rộng rãi trong các ứng dụng web và app mobile. MVC được sử dụng để tách biệt các thành phần của ứng dụng thành ba lớp chức năng riêng biệt:

- **Model:** Lớp này chứa dữ liệu và logic xử lý dữ liệu.
- **View:** Lớp này hiển thị dữ liệu cho người dùng.
- **Controller:** Lớp này xử lý các yêu cầu từ người dùng và cập nhật dữ liệu cho Model và View.



**Hình 4.2:** Mô hình MVC

Mục đích của việc sử dụng kiến trúc MVC là để tách biệt các thành phần của ứng dụng thành ba lớp chức năng riêng biệt, giúp cho việc phát triển, bảo trì và mở rộng ứng dụng được dễ dàng hơn. Một số ưu điểm của mô hình MVC:

- Tổ chức mã nguồn tốt hơn: Giúp cho việc quản lý mã nguồn một cách dễ dàng hơn, giảm thiểu sự phụ thuộc giữa các thành phần.
- Tái sử dụng mã nguồn: Các thành phần có thể được sử dụng lại trong nhiều ứng dụng khác nhau.
- Dễ dàng mở rộng: Dễ dàng thêm hoặc chỉnh sửa các chức năng mà không cần phải thay đổi các thành phần khác.

Trong ứng dụng này, phía client gồm các thành phần chính như sau:

- screens: Gồm các màn hình chính được sử dụng trong ứng dụng như: login\_screen, home\_screen, profile\_screen...
- service: Chứa các logic xử lý của ứng dụng.
- utils: Chứa các hàm hỗ trợ cho ứng dụng.
- widget: Chứa các widget dùng chung trong các screen như: card, list\_item
- assets: Chứa các tài nguyên như hình ảnh, icon, ...

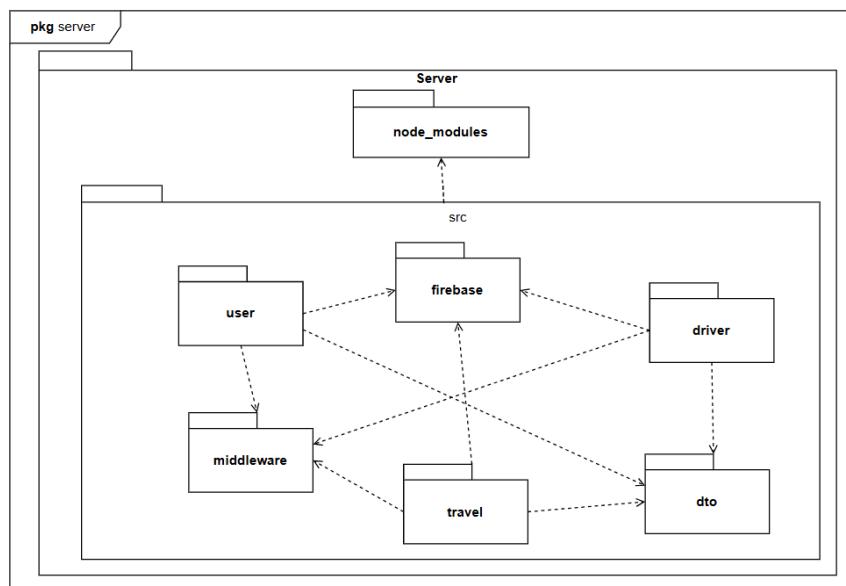
Về phía server, ứng dụng được xây dựng trên framework NestJS, gồm các thành phần chính như sau:

- module: nhóm các chức năng chính của ứng dụng như controller, service, dto...
- controller: Nhận, xử lý yêu cầu và gửi phản hồi lại cho người dùng. Đây là phần Controller trong mô hình MVC. Máy khách chịu trách nhiệm là phần View

trong mô hình MVC. Một số controller của ứng dụng như: travel.controller, user.controller, driver.controller

- service: Chịu trách nhiệm xử lý các logic nghiệp vụ được sử dụng bởi controller hoặc các service khác.
- middleware: Xử lý yêu cầu trước khi nó được gửi đến controller

#### 4.1.2 Thiết kế tổng quan

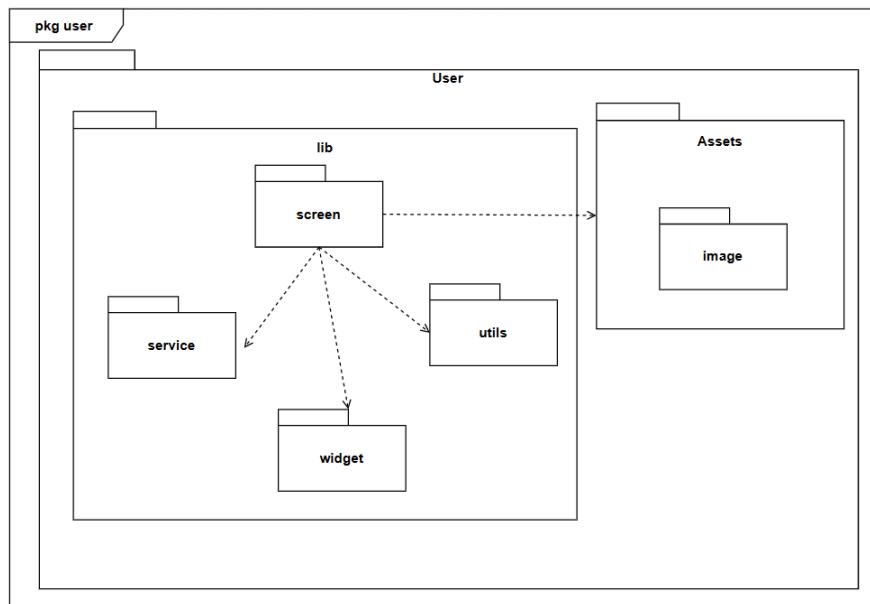


**Hình 4.3:** Biểu đồ phụ thuộc gói Server

Hình 4.3 mô tả cấu trúc gói tổng thể phía server của hệ thống. Tất cả các phụ thuộc được mô tả bằng mũi tên như trong hình.

Server bao gồm các gói sau:

- node-modules: Chứa tất cả các module và thư viện mà server cần sử dụng.
- firebase: Định nghĩa các module, service giúp kết nối và tương tác với firebase.
- user: Quản lý người dùng của ứng dụng.
- driver: Quản lý tài xế của ứng dụng.
- travel: Quản lý các chuyến đi của người dùng.
- dto: Định nghĩa các đối tượng truyền dữ liệu.
- middleware: Xử lý các yêu cầu trước khi nó vào các controller tương ứng



**Hình 4.4:** Biểu đồ phụ thuộc gói Driver

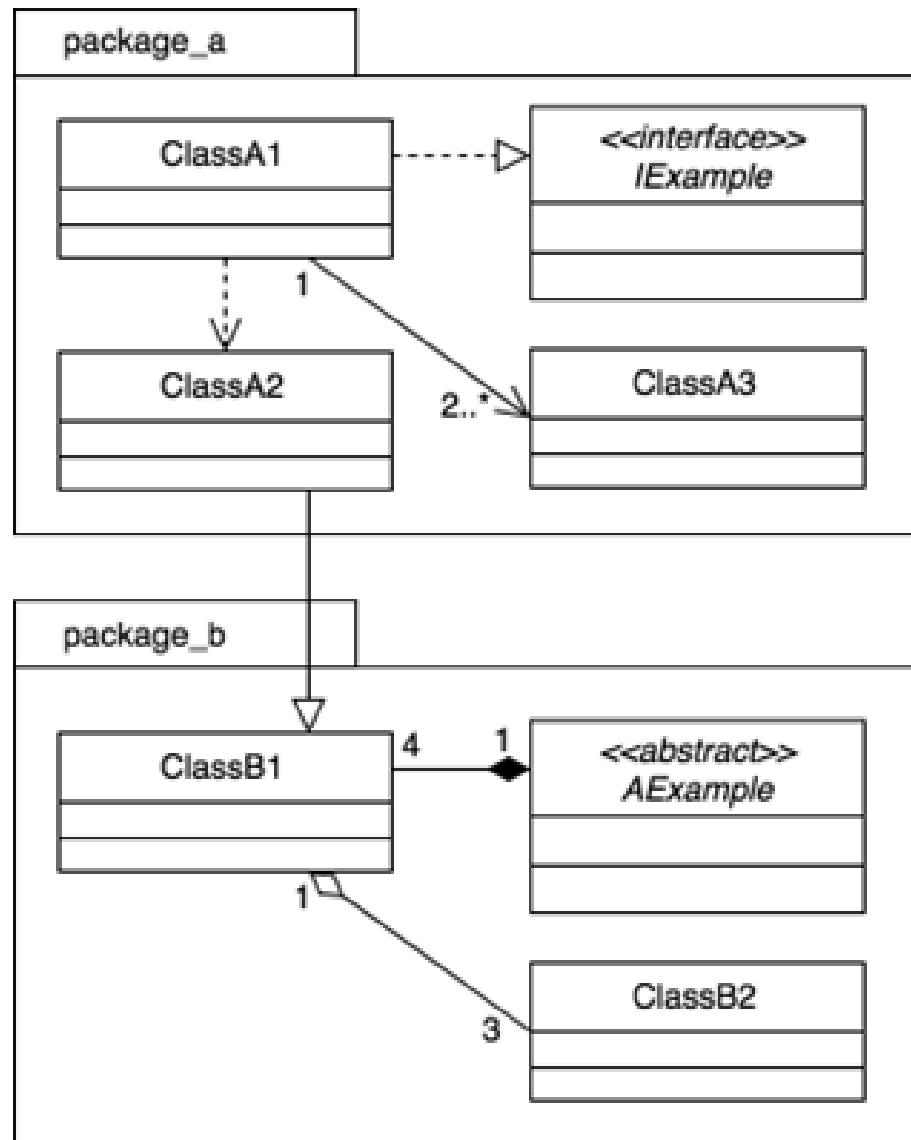
Hình 4.4 mô tả cấu trúc gói phía user và driver của ứng dụng. Tất cả các phụ thuộc được mô tả bằng mũi tên như trong hình.

- assets: Chứa các tài nguyên như hình ảnh, icon, ...
- utils: Chứa các hàm hỗ trợ cho ứng dụng.
- widget: Chứa các widget dùng chung trong các screen như: loading, card...
- screens: Gồm các màn hình chính được sử dụng trong ứng dụng như: login\_screen, home\_screen, profile\_screen...
- service: Chứa các hàm xử lý logic của ứng dụng.

#### 4.1.3 Thiết kế chi tiết gói

Sinh viên thiết kế và lần lượt vẽ biểu đồ thiết kế cho từng package, hoặc một nhóm các package liên quan để giải quyết một vấn đề gì đó. Khi vẽ thiết kế gói, sinh viên chỉ cần đưa tên lớp, không cần chỉ ra các thành viên phương thức và thuộc tính. SV tham khảo ví dụ minh họa trong Hình 4.5.

Sinh viên cần vẽ rõ ràng quan hệ giữa các lớp trong biểu đồ. Các quan hệ bao gồm: phụ thuộc (dependency), kết hợp (association), kết tập (aggregation), hợp thành (composition), kế thừa (inheritance), và thực thi (implementation). Các quan hệ này đều đã được minh họa trong 4.5.

**Hình 4.5:** Ví dụ thiết kế gói

## 4.2 Thiết kế chi tiết

### 4.2.1 Thiết kế giao diện

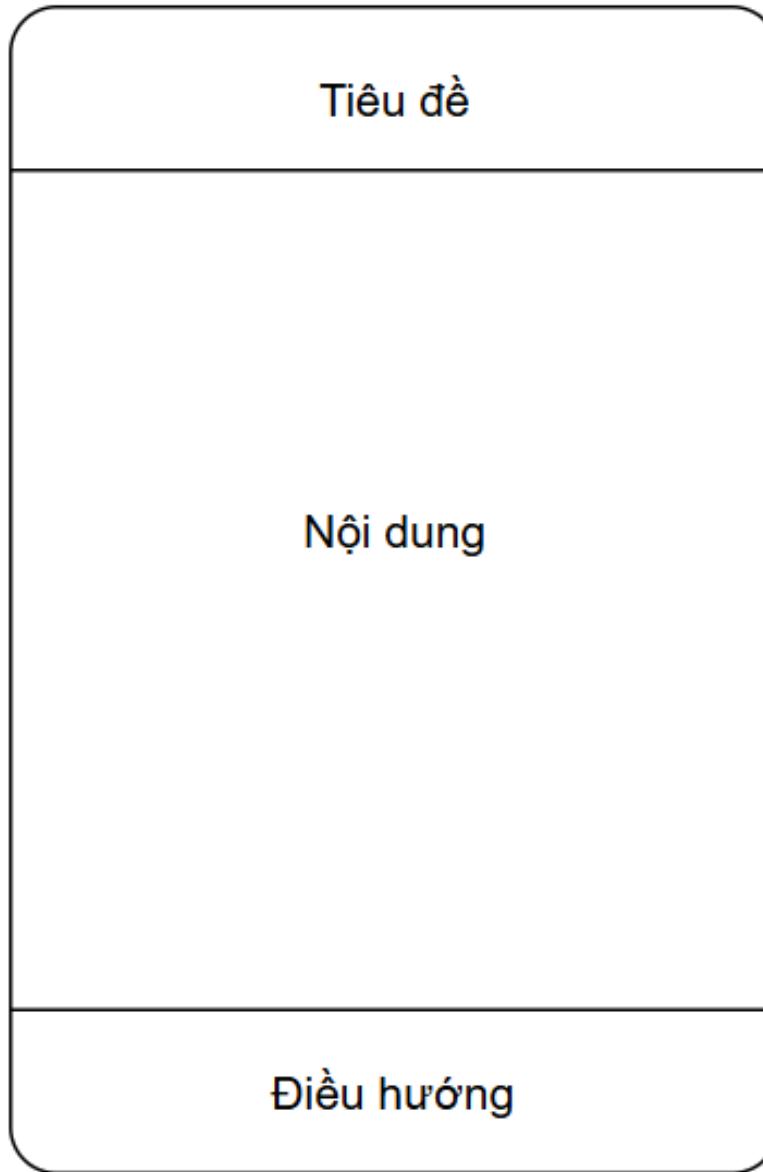
Giao diện của ứng dụng này được thiết kế để sử dụng trên điện thoại Android. Bộ cục các thành phần trong các màn hình ứng dụng được thiết kế đơn giản và dễ sử dụng. Người dùng có thể sử dụng ứng dụng mà không cần đọc bất kỳ hướng dẫn nào.

Thiết kế của ứng dụng cũng giảm thiểu việc sử dụng nhiều màu sắc khác nhau. Thông thường, mỗi trang giao diện chỉ sử dụng 2-3 màu chính (đỏ, xanh lá, xanh dương) để tránh việc gây khó chịu đối với người dùng. Ngoài ra việc sử dụng màu sắc cũng phản ánh tác dụng của chúng. Ví dụ: màu đỏ phản ánh việc xóa, hủy trong khi đó màu xanh lá phản ánh việc thêm mới, xác nhận.

Hầu hết màn hình có 3 phần chính gồm: phần tiêu đề ở trên cùng, phần nội dung

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

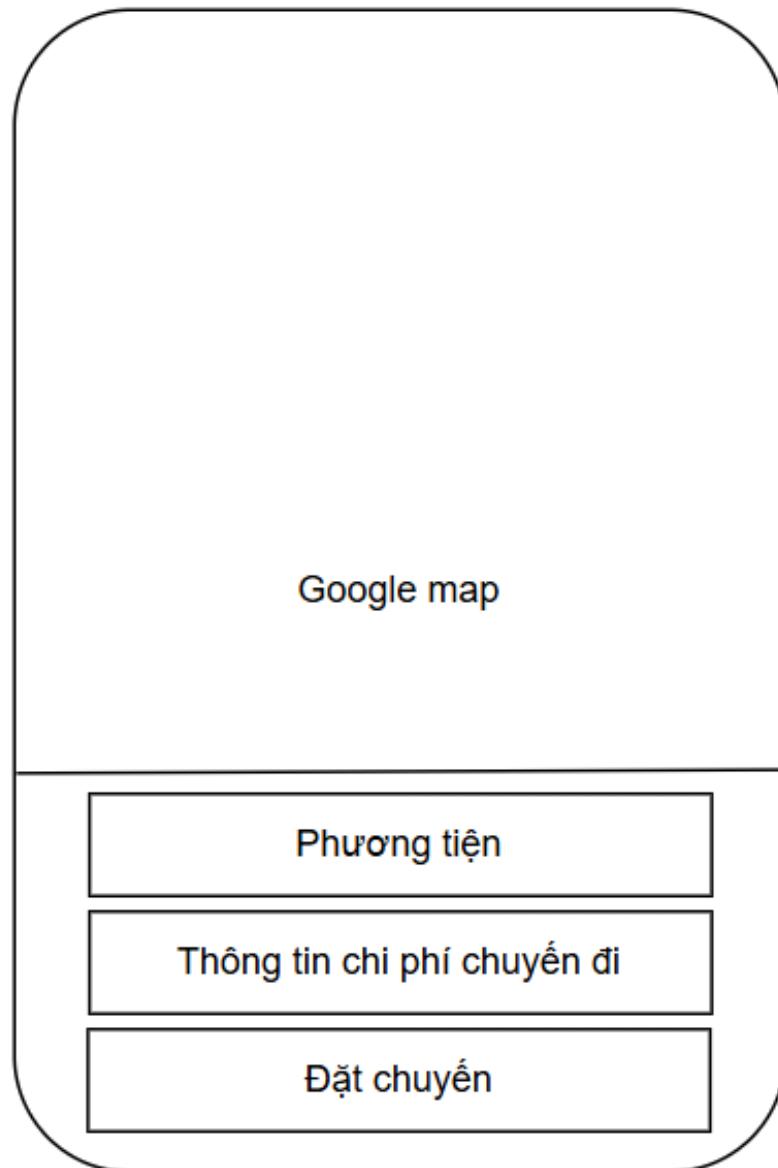
ở giữa và phần điều hướng sang các màn khác ở dưới cùng.



**Hình 4.6:** Thiết kế chính các màn hình

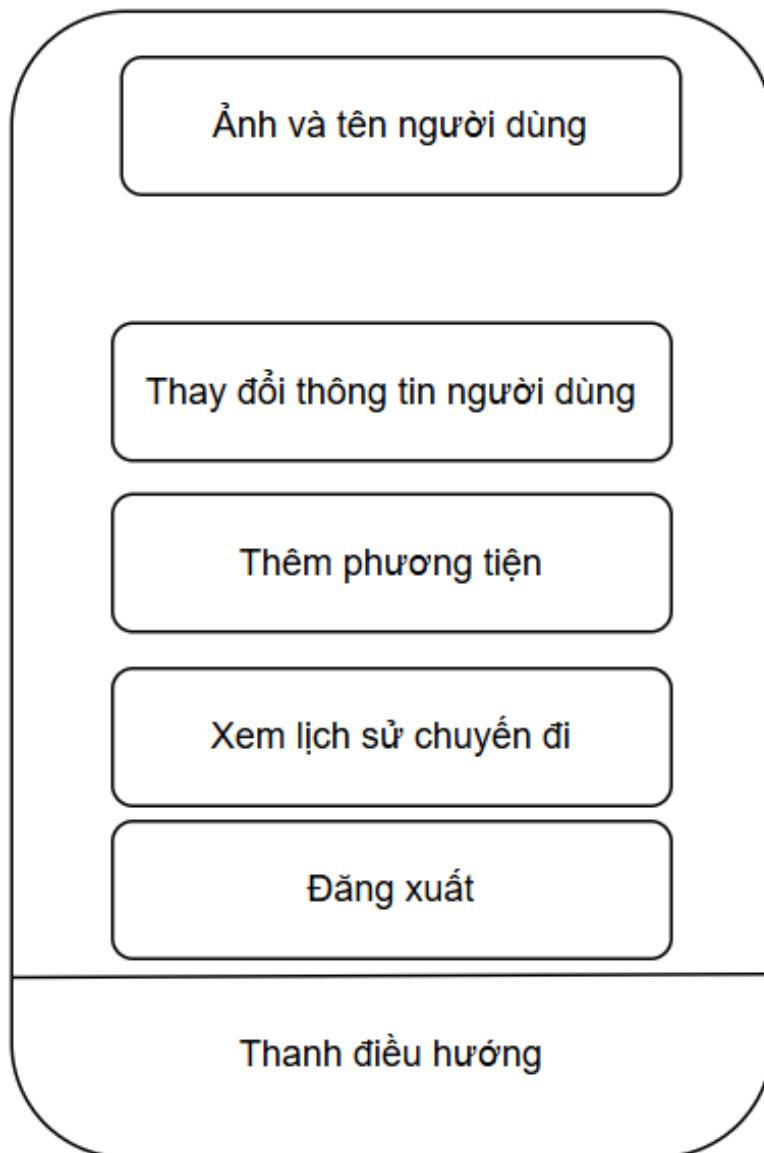
Hình 4.6 mô tả thiết kế chính của các màn hình trong ứng dụng. Phần tiêu đề mô tả để biết đó đang là màn hình nào. Phần nội dung nằm ở giữa và chiếm phần lớn nhất. Phần điều hướng nằm ở dưới dùng chứa các nút để chuyển qua các màn hình khác.

Giao diện Đặt tài xế:



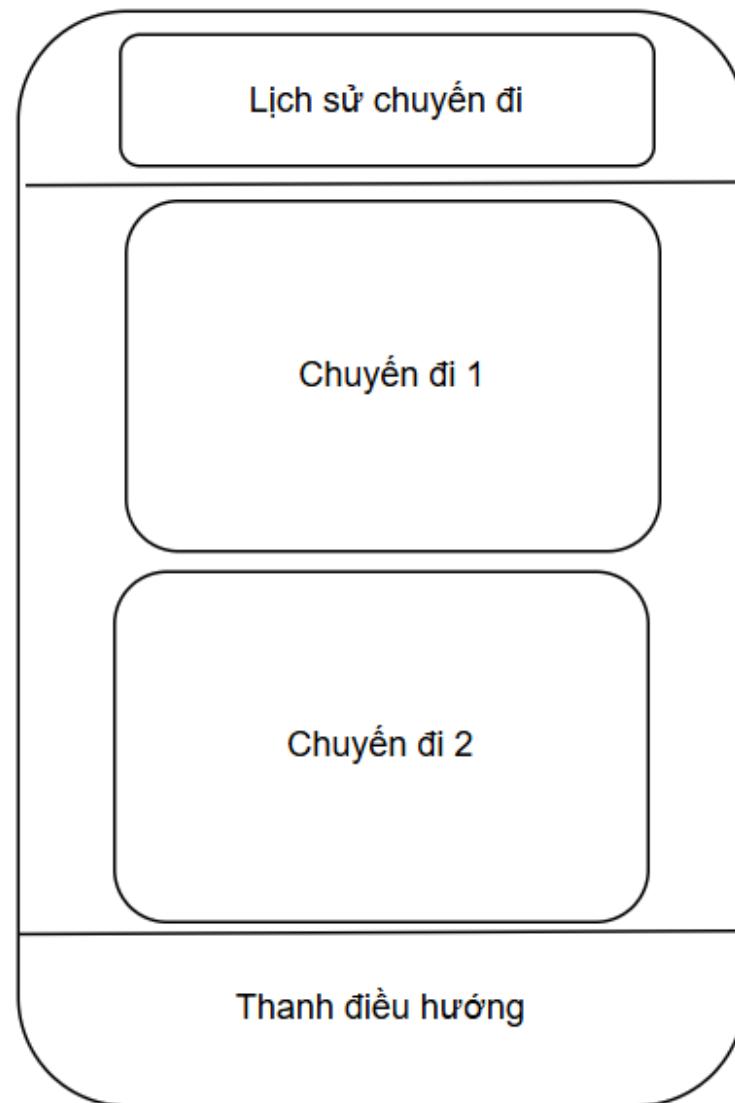
**Hình 4.7:** Thiết kế màn hình Đặt tài xế

Giao diện Thông tin người dùng:



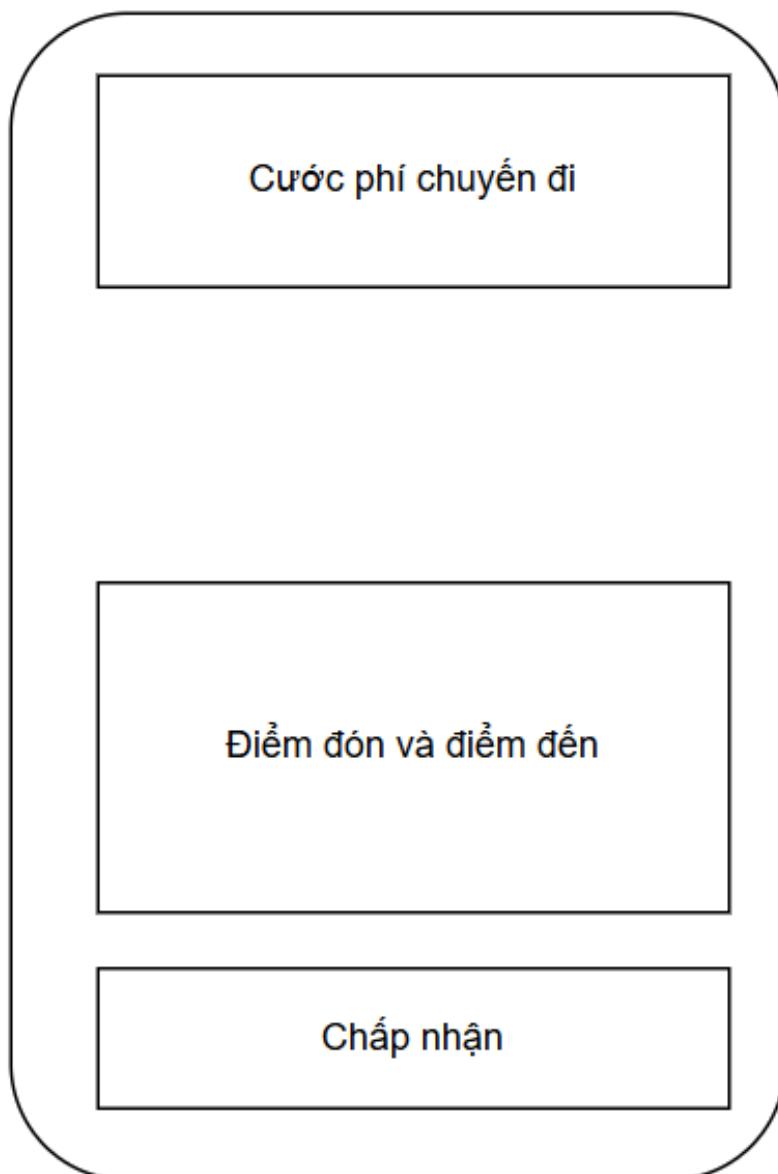
**Hình 4.8:** Thiết kế màn hình Thông tin người dùng

Giao diện Lịch sử chuyến đi của người dùng:



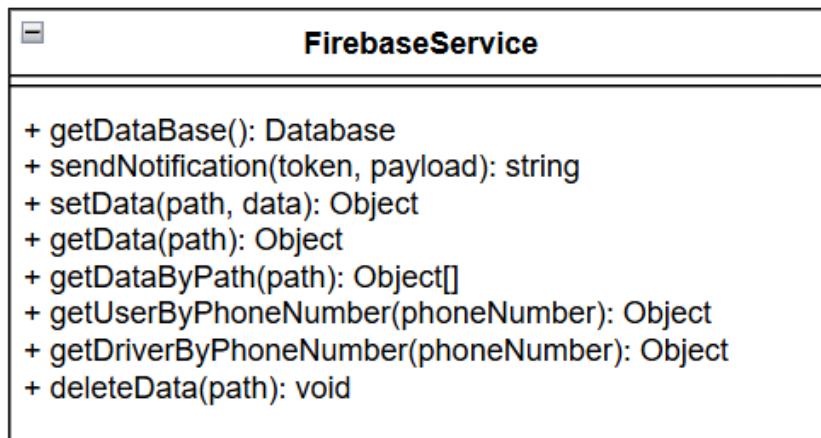
**Hình 4.9:** Thiết kế màn hình Lịch sử chuyến đi của người dùng

Giao diện Thông báo có chuyến đi mới của tài xế:



**Hình 4.10:** Thiết kế màn hình Thông báo có chuyến đi mới của tài xế

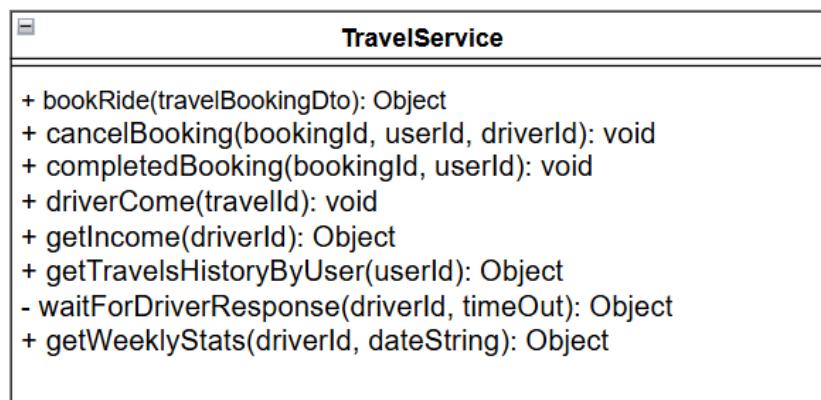
#### 4.2.2 Thiết kế lớp



**Hình 4.11:** Thiết kế lớp FirebaseService

Mô tả chi tiết các phương thức của lớp FirebaseService trong hình 4.11:

- `getDatabase()`: Lấy ra database hiện tại
- `sendNotification(token, payload)`: Gửi thông báo bằng Firebase.
- `setData(path, data)`: Thêm mới hoặc cập nhật dữ liệu.
- `getData(path)`: Lấy dữ liệu theo đường dẫn.
- `getDataByPath(path)`: Lấy dữ liệu theo đường dẫn và trả về mảng các đối tượng.
- `getUserByPhoneNumber(phoneNumber)`: Lấy người dùng theo số điện thoại.
- `getDriverByPhoneNumber(phoneNumber)`: Lấy tài xế theo số điện thoại.
- `deleteData(path)`: Xóa dữ liệu khỏi database theo đường dẫn.

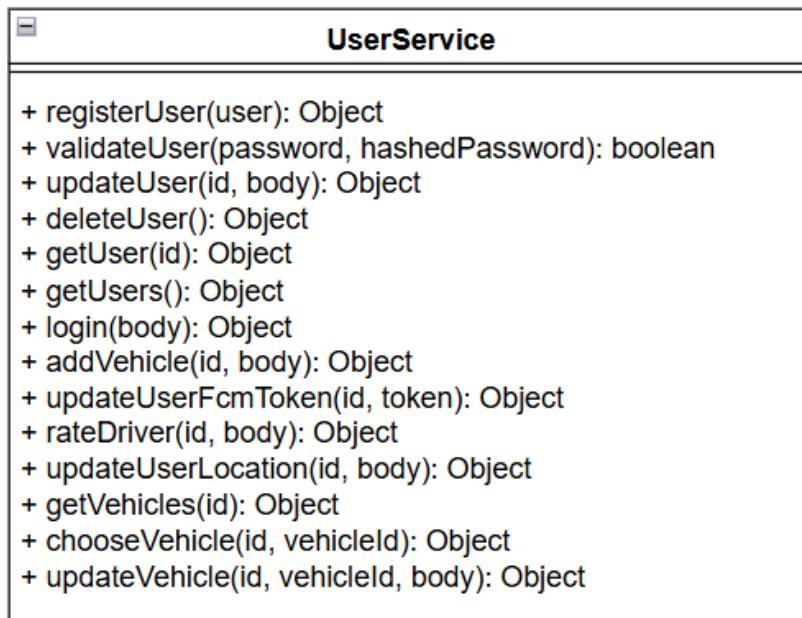


**Hình 4.12:** Thiết kế lớp TravelService

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

Hình 4.12 mô tả các phương thức của lớp TravelService gồm:

- bookRide(travelBookingDto): Tìm kiếm tài xế phù hợp, gửi thông báo cho tài xế rồi chờ phản hồi và gửi lại thông báo cho người dùng.
- cancelBooking(bookingId, userId, driverId): Hủy chuyến đi hiện tại.
- completedBooking(bookingId, userId): Hoàn thành chuyến đi hiện tại.
- driverCome(travelId): Thông báo là tài xế đã đến điểm đón.
- getIncome(driverId): Tính toán tổng thu nhập của tài xế trong 1 ngày.
- getTravelsHistoryByUser(userId): Lấy danh sách các chuyến đi của 1 người dùng.
- waitForDriverResponse(driverId, timeOut): Chờ phản hồi của tài xế về việc nhận chuyến đi.
- getWeeklyStats(driverId, dateString): Lấy tất cả các chuyến đi trong tuần hiện tại của tài xế.

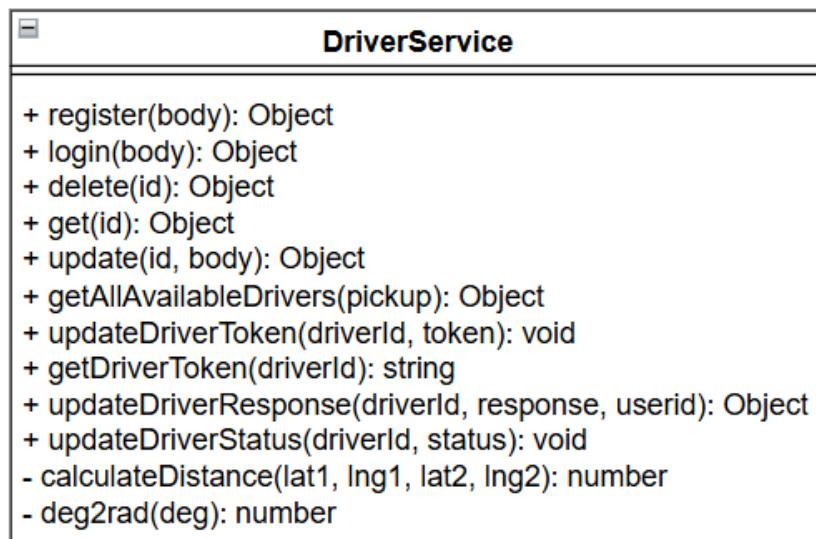


**Hình 4.13:** Thiết kế lớp UserService

Mô tả chi tiết các phương thức có trong lớp UserService ở trong hình 4.13:

- registerUser(user): Thêm người dùng mới.
- validateUser(password, hashedPassword): Kiểm tra mật khẩu người dùng.
- updateUser(id, body): Cập nhật thông tin người dùng.

- deleteUser(id): Xóa người dùng ra khỏi database.
- getUser(id): Lấy thông tin người dùng.
- login(body): Đăng nhập vào ứng dụng.
- addVehicle(id, body): Thêm phương tiện cần lái hộ của người dùng.
- updateUserFcmToken(id, token): Cập nhật FCM token của người dùng.
- rateDriver(id, body): Đánh giá tài xế sau chuyến đi.
- updateUserLocation(id, body): Cập nhật vị trí của người dùng.
- getVehicles(id): Lấy các phương tiện của người dùng.
- chooseVehicle(id, vehicleId): Chọn phương tiện người dùng cần lái hộ.
- getVehicle(id): Lấy thông tin phương tiện mà người dùng đang cần lái hộ.
- updateVehicle(id, vehicleId, body): Cập nhật thông tin phương tiện của người dùng.



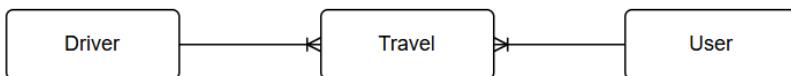
**Hình 4.14:** Thiết kế lớp DriverService

Mô tả chi tiết các phương thức có trong lớp DriverService ở trong hình 4.14:

- register(body): Đăng ký tài xế.
- login(body): Đăng nhập.
- delete(id): Xóa tài khoản.
- get(id): Lấy thông tin tài xế.

- update(id): Cập nhật thông tin tài xế.
- getAllAvailableDrivers(pickup): Lấy danh sách tài xế sẵn sàng nhận chuyến.
- updateDriverToken(driverId, token): Cập nhật FCM token của tài xế.
- getDriverToken(driverId): Lấy FCM token của người dùng.
- updateDriverResponse(driverId, response, userId): Cập nhật trạng thái đồng ý nhận chuyến của tài xế.
- updateDriverStatus(driverId, status): Cập nhật trạng thái sẵn sàng nhận chuyến của tài xế.
- calculateDistance(lat1, lng1, lat2, lng2): Tính khoảng cách giữa 2 địa điểm
- deg2rad(deg): Chuyển đổi từ số đo độ sang radian.

#### **4.2.3 Thiết kế cơ sở dữ liệu**



**Hình 4.15:** Biểu đồ thực thể liên kết

| <b>Thực thể</b> | <b>Giải thích</b>              |
|-----------------|--------------------------------|
| User            | Lưu thông tin người dùng       |
| Driver          | Lưu thông tin tài xế           |
| Travel          | Lưu thông tin về các chuyến đi |

**Bảng 4.1:** Giải thích về các thực thể

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

---

| Tên cột                    | Loại dữ liệu | Ý nghĩa                               |
|----------------------------|--------------|---------------------------------------|
| avatar                     | String       | Ảnh đại diện của tài xế               |
| avgRate                    | Number       | Đánh giá trung bình                   |
| citizenIdentificationFront | String       | Ảnh mặt trước CCCD                    |
| citizenIdentificationBack  | String       | Ảnh mặt sau CCCD                      |
| drivingLicense             | String       | Ảnh giấy phép lái xe                  |
| fcmToken                   | String       | FCM token nhận thông báo của Firebase |
| fullName                   | String       | Họ tên đầy đủ                         |
| location                   | Object       | Vị trí hiện tại của tài xế            |
| password                   | String       | Mật khẩu                              |
| phoneNumber                | String       | Số điện thoại                         |
| rateDriver                 | Number[]     | Các đánh giá của người dùng           |
| response                   | String       | Trạng thái đồng ý nhận cuốc           |
| status                     | String       | Trạng thái hiện tại của tài xế        |

**Bảng 4.2:** Thiết kế chi tiết thực thể tài xế

| Tên cột             | Loại dữ liệu | Ý nghĩa   |
|---------------------|--------------|---|
| destinationLocation | Object       | Vị trí điểm đến                                       |
| destinationString   | String       | Điểm đến  |
| distance            | Number       | Khoảng cách   |
| driverId            | String       | ID tài xế   |
| duration            | Number       | Thời gian dự kiến khi không có phương tiện trên đường |
| durationInTraffic   | Number       | Thời gian dự kiến khi có phương tiện trên đường       |
| fare                | Number       | Giá cước  |
| id                  | String       | Id chuyến đi  |
| pickupLocation      | Number       | Vị trí điểm đón                                       |
| pickupString        | String       | Điểm đón  |
| poliline            | Object[]     | Đường đi đề xuất                                      |
| status              | String       | Trạng thái chuyến đi                                  |
| timeEnd             | String       | Thời gian kết thúc chuyến đi                          |
| timeStart           | String       | Thời gian bắt đầu chuyến đi                           |
| userId              | String       | ID người dùng   |

**Bảng 4.3:** Thiết kế chi tiết thực thể chuyến đi

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

| Tên cột      | Loại dữ liệu | Ý nghĩa                                  |
|--------------|--------------|--|
| email        | String       | Email của người dùng                     |
| fcmToken     | String       | FCM token để nhận thông báo của Firebase |
| fullName     | String       | Họ tên của người dùng                    |
| id           | String       | ID người dùng                            |
| password     | String       | Mật khẩu                                 |
| phoneNumber  | String       | Số điện thoại của người dùng             |
| userAvatar   | String       | Ảnh đại diện của người dùng              |
| userLocation | Object       | Vị trí của người dùng                    |
| vehicles     | Object[]     | Danh sách các phương tiện của người dùng |

Bảng 4.4: Thiết kế chi tiết thực thể người dùng

### 4.3 Xây dựng ứng dụng

#### 4.3.1 Thư viện và công cụ sử dụng

| Mục đích           | Công cụ                    | Địa chỉ URL   |
|--------------------|----------------------------|---|
| IDE lập trình      | Visual Studio Code         | <a href="https://code.visualstudio.com/">https://code.visualstudio.com/</a>     |
| Version Control    | Github                     | <a href="https://github.com/">https://github.com/</a>                           |
| Database           | Firebase Realtime Database | <a href="https://firebase.google.com/">https://firebase.google.com/</a>         |
| Môi trường backend | NodeJS                     | <a href="https://nodejs.org/en">https://nodejs.org/en</a>                       |
| Framework backend  | NestJS                     | <a href="https://nestjs.com/">https://nestjs.com/</a>                           |
| Ngôn ngữ frontend  | Dart                       | <a href="https://dart.dev/">https://dart.dev/</a>                               |
| Framework frontend | Flutter                    | <a href="https://docs.flutter.dev/">https://docs.flutter.dev/</a>               |
| Lưu trữ files      | Firebase Storage           | <a href="https://firebase.google.com/">https://firebase.google.com/</a>         |
| Tìm kiếm địa chỉ   | Google Map Platform        | <a href="https://mapsplatform.google.com/">https://mapsplatform.google.com/</a> |

Bảng 4.5: Danh sách thư viện và công cụ sử dụng

#### 4.3.2 Kết quả đạt được

Sau quá trình nghiên cứu, tìm hiểu và phát triển ứng dụng ViSafe BK, em đã triển khai bản APK phiên bản 1.0.0 chạy trên nền tảng Android. Ứng dụng có các chức năng chính như: tìm kiếm và đặt tài xế, xem thông tin về các chuyến đi. Thông tin chi tiết về ứng dụng được trình bày trong bảng 4.6:

| STT | Thông tin                              | Số liệu |
|-----|--|---------|
| 1   | Số gói trong ứng dụng tài xế           | 4       |
| 2   | Số dòng code trong ứng dụng tài xế     | 2880    |
| 3   | Số file trong ứng dụng tài xế          | 20      |
| 4   | Số gói trong ứng dụng người dùng       | 4       |
| 5   | Số dòng code trong ứng dụng người dùng | 3370    |
| 6   | Số file trong ứng dụng người dùng      | 20      |
| 7   | Số gói trong ứng dụng backend          | 8       |
| 8   | Số dòng code trong ứng dụng backend    | 2570    |
| 9   | Số file trong ứng dụng backend         | 38      |
| 10  | Số document trong firebase             | 3       |

**Bảng 4.6:** Thống kê thông tin ứng dụng

#### 4.3.3 Minh họa các chức năng chính

Sau đây là màn hình các chức năng chính, quan trọng trong ứng dụng:

Giao diện màn hình Thông báo chuyển đi:



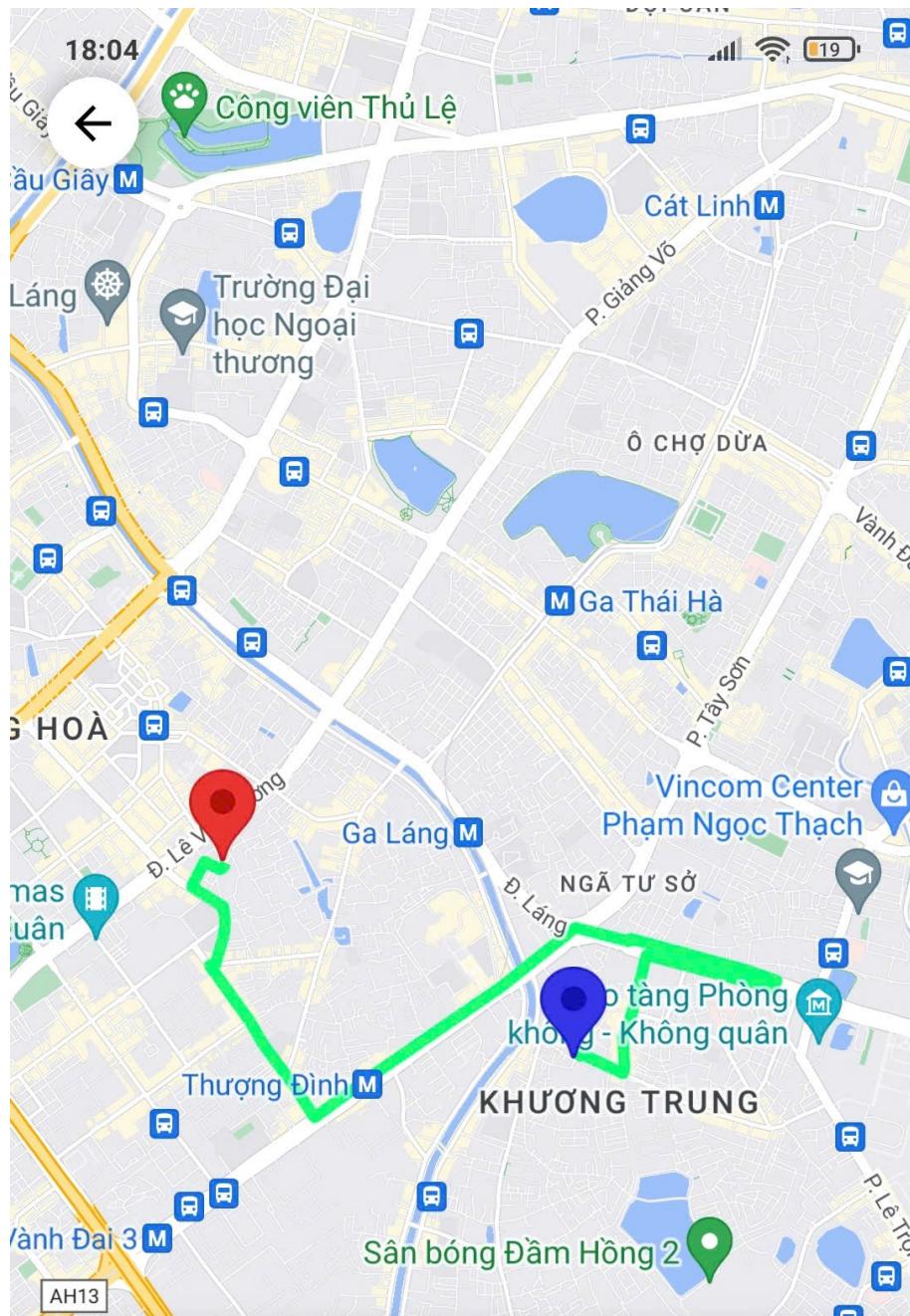
Hình 4.16: Màn hình thông báo chuyến đi của tài xế

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

Màn hình thông báo chuyến đi cung cấp cho tài xế biết thông tin về giá của chuyến đi, điểm đón và điểm đến. Tài xế có 10 giây để quyết định chấp nhận hoặc không chấp nhận chuyến xe.

Giao diện màn hình Tìm đặt tài xế:

#### CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG



Phương tiện của tôi:  
s300

**Thay đổi**

Giá cước:

**60.368 đ**

Tiền mặt

Khoảng cách:

**4.627 km**

**Đặt Tài Xe**

Hình 4.17: Màn hình Tìm đặt tài xế của người dùng

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

Người dùng biết được chi phí phải trả cho chuyến đi, khoảng cách, tuyến đường đi dự kiến giữa 2 địa điểm. Ngoài ra người dùng phải chọn phương tiện cần lái hộ trước khi đặt tài xế. Sau khi có đầy đủ thông tin, người dùng nhấn "Đặt tài xế" để tìm tài xế thích hợp khi có nhu cầu thuê lái hộ.

Giao diện màn hình Lịch sử chuyến đi của người dùng:

18:05     

## Lịch sử chuyến đi

**Điểm đón: 102 Phố Tô Vĩnh Diện, Khương Trung, Thanh Xuân, Hà Nội, Việt Nam**

Điểm đến: 21 Đường Lê Văn Lương, Nhân Chính, Thanh Xuân, Hà Nội, Việt Nam

**Trạng thái: Hoàn thành**

Thời gian bắt đầu: 27/12/2024 17:12

Thời gian kết thúc: 27/12/2024 17:17

Quãng đường: 4.627 km

Chi phí: 69.952 VND

**Điểm đón: 102 Đường Trường Chinh, Phương Mai, Đống Đa, Hà Nội, Việt Nam**

Điểm đến: 156 Đường Tân Triều, Triều Khúc, Tân Triều, Hà Đông, Hà Nội, Việt Nam

**Trạng thái: Hoàn thành**

Thời gian bắt đầu: 27/12/2024 17:02

Thời gian kết thúc: 27/12/2024 17:03

Quãng đường: 7.082 km

Chi phí: 82.860 VND

**Điểm đón: Bách Khoa, Hai Bà Trưng, Hà Nội, Việt Nam**

Điểm đến: Hoàng Đạo Thuý, Trung Hòa Nhân Chính, Nhân Chính, Thanh Xuân, Hà Nội, Việt Nam

**Trạng thái: Hoàn thành**

Thời gian bắt đầu: 24/12/2024 10:33

Thời gian kết thúc: 24/12/2024 10:33

Trang chủ

Hoạt động

Tài khoản

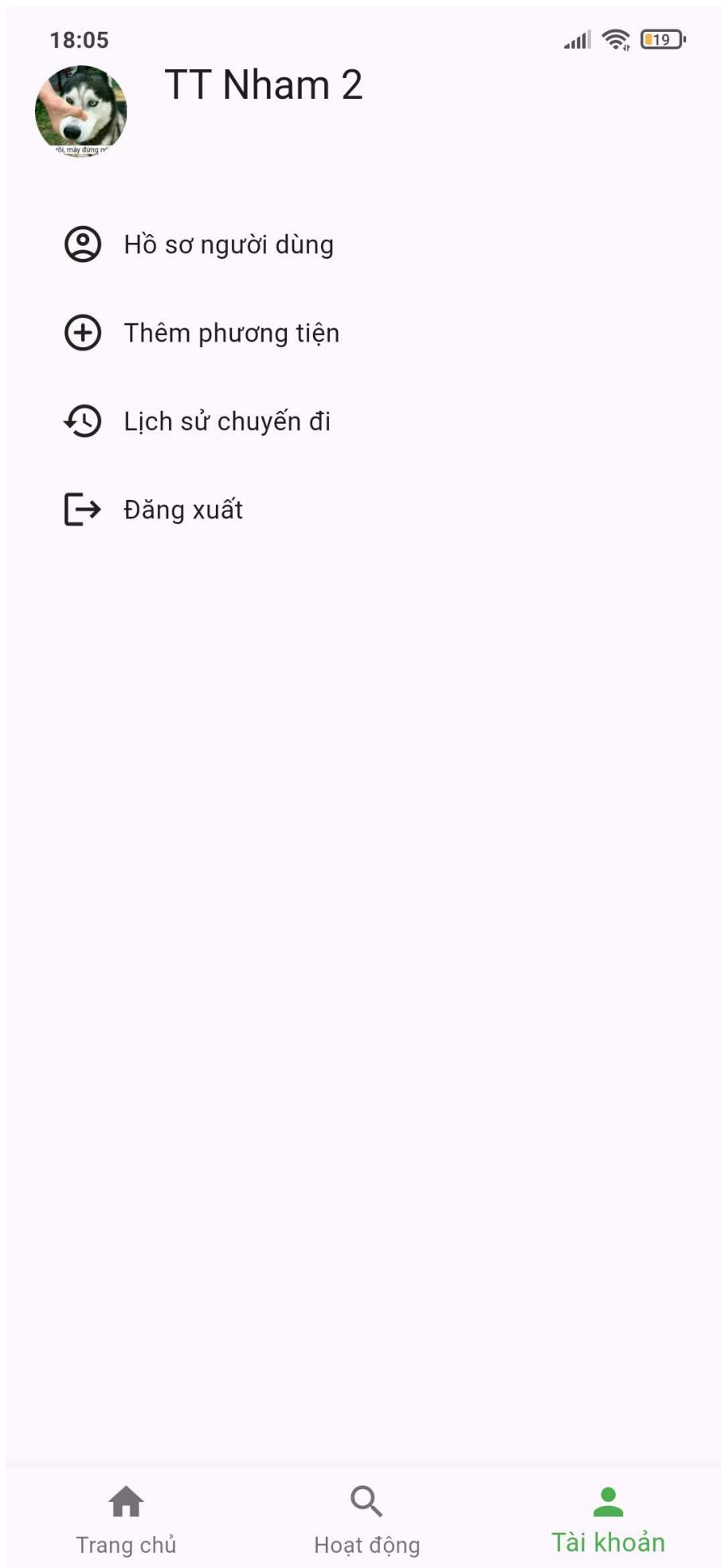
Hình 4.18: Màn hình Lịch sử chuyến đi của người dùng

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

Khi nhấn vào phần hoạt động trên app của người dùng thì phần lịch sử hoạt động này sẽ hiện ra. Nó cho người dùng biết rằng họ đã thực hiện những chuyến đi nào và thông tin về những chuyến đi đó.

Giao diện màn hình Tài khoản người dùng:

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

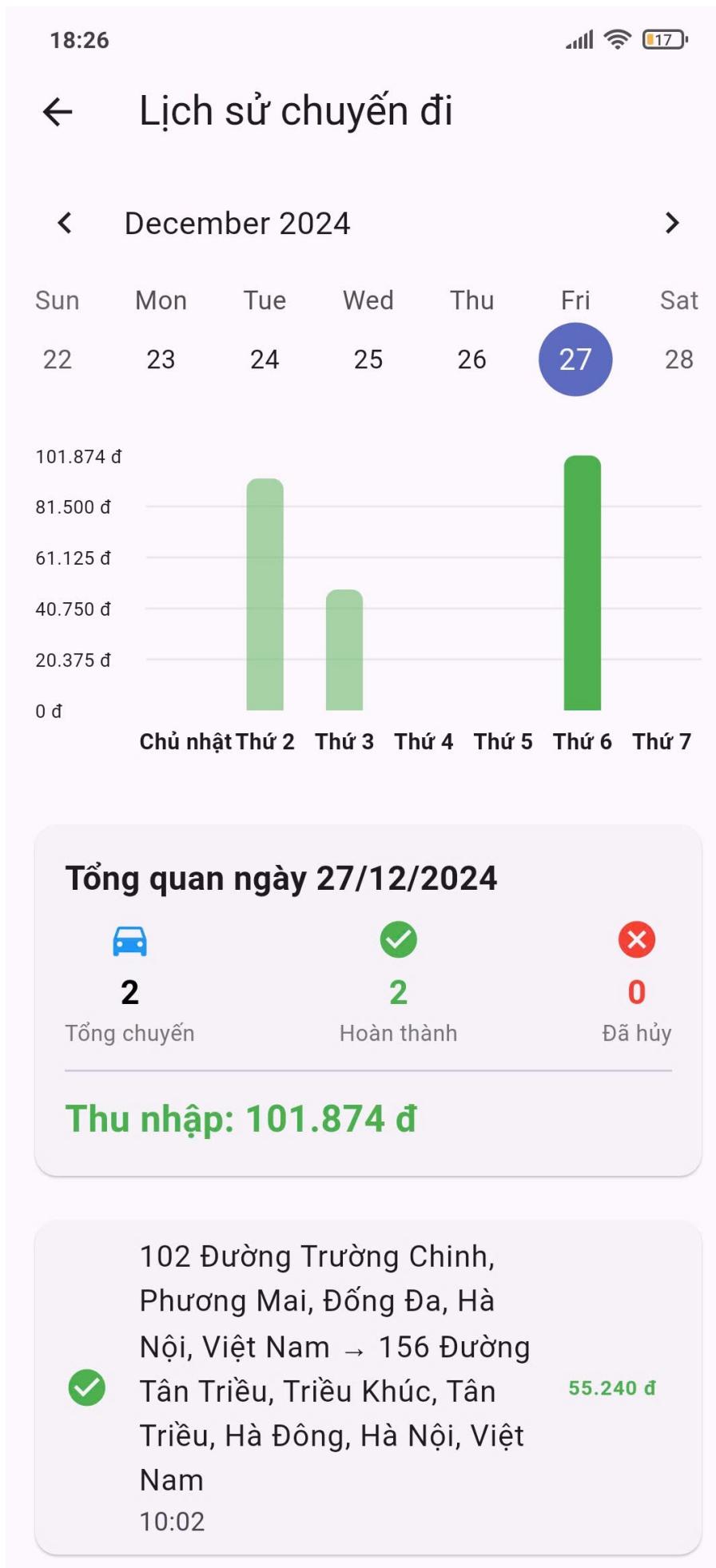


**Hình 4.19:** Màn hình Tài khoản người dùng

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

Sau khi người dùng đăng nhập vào ứng dụng có thể nhấn vào phần "Tài khoản" để xem thông tin về tài khoản của họ. Người dùng có thể thay đổi thông tin bằng cách nhấn vào phần "Hồ sơ người dùng". Thêm phương tiện muốn lái hộ bằng cách nhấn vào phần "Thêm phương tiện" hoặc muốn thoát ứng dụng thì nhấn vào phần "Đăng xuất".

Giao diện màn hình Lịch sử chuyến đi của tài xế:



Hình 4.20: Màn hình Lịch sử chuyến đi của tài xế

Sau khi tài xế đăng nhập, ở màn hình chính, tài xế bấm vào nút đăng nhập để xem thống kê thu nhập và chuyến đi trong ngày. Thu nhập được thống kê bằng đồ thị theo các ngày trong tuần. Tài xế có thể thay đổi ngày để xem lịch sử chuyến đi của các ngày khác.

#### **4.4 Kiểm thử**

2-3 trang 2-3 chuc nang

##### **4.4.1 Kiểm thử đăng nhập người dùng**

Dữ liệu đầu vào:

- phoneNumber: số điện thoại của người dùng.
- password: mật khẩu của người dùng.

Kết quả trả về:

- False: Đăng nhập thất bại, đưa ra lý do đăng nhập thất bại.
- True: Đăng nhập thành công, chuyển màn hình sang phần màn hình chính.

| STT | Dữ liệu vào                                   | Đầu ra cần đạt                                   |
|-----|---|--|
| 1   | phoneNumber: null, password:                  | False (Số điện thoại và mật khẩu null)           |
| 2   | phoneNumber: "", password: "123456"           | False (Số điện thoại không đúng định dạng)       |
| 3   | phoneNumber: "0123456789", password: ""       | False (Mật khẩu phải có ít nhất 6 ký tự)         |
| 4   | phoneNumber: "0123456789", password: "000000" | False (Số điện thoại hoặc mật khẩu không hợp lệ) |
| 5   | phoneNumber: "0372356956", password: "123456" | True (Chuyển sang màn hình chính)                |

**Bảng 4.7:** Kiểm thử đăng nhập người dùng

##### **4.4.2 Kiểm thử tính chi phí chuyến đi**

Dữ liệu đầu vào:

- pickup: tọa độ điểm đón.
- destination: tọa độ điểm đến.

Kết quả trả về:

- False: Đưa ra lý do không tính toán được chi phí.
- True: Chuyển sang màn hình đặt xe để xem thông tin.

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

| STT | Dữ liệu vào   |
|-----|---|
| 1   | pickup: null, destination: null   |
| 2   | pickup: latitude: 20.998944,longitude: 105.806999<br>destination: null                                      |
| 3   | pickup: null,<br>destination: latitude: 21.006348,longitude: 105.806999                                     |
| 4   | pickup: latitude: 20.998944,longitude: 105.806999<br>destination: latitude: 21.006348,longitude: 105.806999 |

**Bảng 4.8:** Kiểm thử tính chi phí chuyến đi

### 4.5 Triển khai

Sinh viên trình bày mô hình và/hoặc cách thức triển khai thử nghiệm/thực tế. Ứng dụng của sinh viên được triển khai trên server/thiết bị gì, cấu hình như thế nào. Kết quả triển khai thử nghiệm nếu có (số lượng người dùng, số lượng truy cập, thời gian phản hồi, phản hồi người dùng, khả năng chịu tải, các thống kê, v.v.)

Trong dự án này, em đã thực hiện build file apk của ứng dụng ViSafe BK:

- Triển khai version 1 với sự trợ giúp của Flutter
- Ứng dụng có thể tải từ các thiết bị Android

Các bước để kiểm tra ứng dụng trên môi trường máy local:

- Bước 1: Cài đặt và cấu hình môi trường cần thiết để chạy thiết bị (Git, NodeJS, Firebase, Flutter).
- Bước 2: Tải mã nguồn từ Github hoặc giải nén từ file zip.
- Bước 3: Chạy lệnh "npm i" trong thư mục server.
- Bước 4: Chạy lệnh "flutter pub get" trong thư mục của ứng dụng user và driver.
- Bước 5: Cấu hình biến môi trường cho môi trường server.
- Bước 6: Chạy lệnh "npm run start:dev" để chạy server.
- Bước 7: Thay đổi đường dẫn của api trong cấu hình ứng dụng driver và user. Nếu chạy trên máy ảo thì dùng "http://10.0.2.2:3000" hoặc dùng máy vật lý thì sẽ dùng địa chỉ ipv4 của máy chạy server.
- Bước 8: Dùng lệnh "flutter run" để chạy ứng dụng user và driver.

## CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

### 5.1 Lưu trữ file

#### 5.1.1 Vấn đề đặt ra

Với các chức năng sử dụng hình ảnh như: đăng ký tài xế, cập nhật thông tin hay xem các phương tiện cần lái hộ sẽ cần phải lưu trữ lại các file hình ảnh. Tuy nhiên việc lưu trữ trực tiếp ảnh trong database sẽ khiến database trở nặng hơn. Do đó, cần phải có nơi để lưu trữ lại những hình ảnh đó để giảm thiểu dung lượng của database.

#### 5.1.2 Giải pháp

Trong dự án này, em sử dụng kho lưu trữ tệp của Firebase đó là Firebase Cloud Storage. Với Firebase Cloud Storage, khi người dùng upload ảnh lên ứng dụng, ảnh sẽ được lưu ở Firebase Cloud Storage và nó sẽ trả về cho người dùng một URL hình ảnh và sử dụng URL đó để hiển thị ảnh cũng như lưu vào database.

Hình 5.1 minh họa việc tải một hình ảnh lên Firebase là lấy URL để hiển thị ảnh đó:



The screenshot shows a mobile application interface with a dark background. In the top left corner, there are three colored dots (red, yellow, green). The main area contains the following Dart code:

```
1 // chọn và upload ảnh
2 Future<void> _pickAndUploadImage() async {
3     final ImagePicker picker = ImagePicker();
4     try {
5         final XFile? image = await picker.pickImage(source: ImageSource.gallery);
6         if (image == null) return;
7
8         setState(() {
9             _imageFile = File(image.path);
10            _isUploading = true;
11        });
12
13        // Upload lên Firebase Storage
14        final storageRef = FirebaseStorage.instance
15            .ref()
16            .child('user_avatars')
17            .child('${DateTime.now().millisecondsSinceEpoch}.jpg');
18
19        await storageRef.putFile(_imageFile!);
20        final downloadUrl = await storageRef.getDownloadURL();
21
22        setState(() {
23            _imageUrl = downloadUrl;
24            _isUploading = false;
25        });
26    } catch (e) {
27        setState(() {
28            _isUploading = false;
29        });
30        print(e);
31        ScaffoldMessenger.of(context).showSnackBar(
32            SnackBar(content: Text('Lỗi khi tải ảnh lên: $e')),
33        );
34    }
35}
```

Hình 5.1: Tải hình ảnh lên Firebase

### 5.1.3 Kết quả

Việc sử dụng Firebase Cloud Storage đã giúp đơn giản hóa việc lưu trữ hình ảnh, đơn giản hóa việc truyền tải và hiển thị ảnh và đồng thời giúp giảm dung lượng lưu trữ không cần thiết trong database. Kết quả của việc sử dụng Firebase Cloud Storage và lưu các URL trong database được hiển thị như trong hình:

```
— email: "trieunham2002@gmail.com"
— fcmToken: "d7uclqm6RUW3v7G_wNhZuT:APA91bExA8w5iSS0N16vYNjq607tqsygRa1PeibFELnkU97dEl1aNJCNs6...
— fullName: "TT Nham 2"
— id: "0096fda3-c818-4f98-a24a-e0b887b128d4"
— password: "$2a$10$C6PkFPhWT9PaD1dhY2p3s0qqQ2T3SgxM8ZjzqgpZgP5o2iU06fG2hs"
— phoneNumber: "0372356956"
— status: "offline"
— userAvatar: "https://firebasestorage.googleapis.com/v0/b/datn-7d31c.appspot.com/o/user_avatars%2F173246...
```

**Hình 5.2:** Hiển thị URL hình ảnh trong database

## 5.2 Thuật toán tính toán độ ưu tiên của tài xế khi thực hiện đặt xe

### 5.2.1 Vấn đề đặt ra

Trong khi người dùng đặt chuyến xe, vào một thời điểm sẽ có nhiều tài xế đang hoạt động và có thể nhận chuyến xe. Do đó, cần phải có một thuật toán để tính toán tài xế tối ưu nhất để nhận chuyến xe đó.

### 5.2.2 Giải pháp

Để tính toán độ ưu tiên của tài xế, em dựa vào các thông tin như sau sau:

- Đánh giá của tài xế (số sao).
- Khoảng cách mà tài xế đi đến địa điểm đón người đặt chuyến.

Trước hết để tính toán được khoảng cách mà tài xế đi đến địa điểm đón người đặt chuyến, em sử dụng công thức Haversine tính khoảng cách giữa hai điểm trên Trái Đất. Việc tính toán khoảng cách được thực hiện như hình 5.3:



```

1  private calculateDistance(
2      lat1: number,
3      lng1: number,
4      lat2: number,
5      lng2: number,
6  ): number {
7      const R = 6371; // Radius of the Earth in km
8      const dLat = this.deg2rad(lat2 - lat1);
9      const dLon = this.deg2rad(lng2 - lng1);
10     const a =
11         Math.sin(dLat / 2) * Math.sin(dLat / 2) +
12         Math.cos(this.deg2rad(lat1)) *
13             Math.cos(this.deg2rad(lat2)) *
14             Math.sin(dLon / 2) *
15             Math.sin(dLon / 2);
16     const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
17     return R * c;
18 }
19
20 private deg2rad(deg: number): number {
21     return deg * (Math.PI / 180);
22 }

```

**Hình 5.3:** Công thức Haversine

Sau khi tính toán được khoảng cách, em sẽ thực hiện việc chuẩn hóa và sắp xếp các tài xế theo độ ưu tiên. Công việc trên được thể hiện như trong hình 5.4:



```

1 // Tính điểm cho mỗi tài xế dựa trên đánh giá và khoảng cách
2 const driversWithScore = drivers.map((driver) => {
3     const distance = driver.distance;
4     const normalizedRating = (driver.rate || 0) / 5;
5     const normalizedDistance = 1 - Math.min(distance, 3000) / 3000;
6     const score = normalizedRating * 0.7 + normalizedDistance * 0.3;
7
8     return {
9         ...driver,
10        score,
11    };
12 });
13
14 // Sắp xếp tài xế theo điểm từ cao xuống thấp
15 const sortedDrivers = driversWithScore.sort(
16     (a, b) => b.score - a.score,
17 );

```

**Hình 5.4:** Sắp xếp các tài xế theo độ ưu tiên

### 5.2.3 Kết quả

Việc thực hiện tính toán khoảng cách và sắp xếp các tài xế theo độ ưu tiên đã giúp cho việc chọn tài xế phù hợp với mỗi người dùng, mỗi chuyến đi được chính xác hơn. Giảm thiểu thời gian chờ đợi cho người đặt xe và tăng độ tin cậy trong việc lái xe.

### 5.3 Theo dõi lịch trình di chuyển của tài xế

#### 5.3.1 Vấn đề đặt ra

Trong quá trình thực hiện chuyến đi, người dùng cần biết được vị trí hiện tại của tài xế để ước tính thời gian chờ đợi cũng như theo dõi hành trình di chuyển. Điều này đặc biệt quan trọng vì:

- Giúp người dùng chủ động trong việc chuẩn bị và sắp xếp thời gian.
- Tăng tính minh bạch và độ tin cậy của dịch vụ.
- Đảm bảo an toàn cho cả người dùng và tài xế thông qua việc giám sát hành trình.
- Hỗ trợ việc giải quyết các khiếu nại hoặc tranh chấp nếu có phát sinh.

#### 5.3.2 Giải pháp

Việc theo dõi vị trí được thực hiện ở cả 2 phía: người đặt xe và tài xế.

**Phía tài xế:** Ứng dụng sử dụng Geolocator để lấy vị trí với độ chính xác cao và đồng thời cập nhật liên tục vào Firebase Realtime Database bằng cách sử dụng StreamBuilder.

Hình 5.5 minh họa việc theo dõi vị trí của tài xế:



```

1 StreamSubscription<Position>? _positionSubscription;
2   final DatabaseReference _databaseRef = FirebaseDatabase.instance.ref();
3
4   void _startLocationTracking() {
5     const LocationSettings locationSettings = LocationSettings(
6       accuracy: LocationAccuracy.high,
7       distanceFilter: 1,
8     );
9
10   _positionSubscription =
11     Geolocator.getPositionStream(locationSettings: locationSettings)
12       .listen((Position position) async {
13       setState(() {
14         currentPosition = LatLng(position.latitude, position.longitude);
15       });
16
17       await _updateLocationToFirebase(position);
18     });
19   }
20
21 Future<void> _updateLocationToFirebase(Position position) async {
22   try {
23     await _databaseRef
24       .child('drivers')
25       .child(widget.data['driverId'])
26       .child('location')
27       .set({
28         'latitude': position.latitude,
29         'longitude': position.longitude,
30       });
31   } catch (e) {
32     print('Error in _updateLocationToFirebase $e');
33     showAlertDialog(context, 'Lỗi', e.toString());
34   }
35 }

```

**Hình 5.5:** Cập nhật thay đổi vị trí của tài xế

**Phía người đặt xe:** Ứng dụng lắng nghe sự thay đổi vị trí của tài xế từ Firebase Realtime Database và cập nhật thay đổi lên giao diện bản đồ của người dùng.

Hình 5.6 minh họa việc theo dõi vị trí của người đặt xe:



```

1 void _initializeDriverTracking() {
2     final driverId = widget.data['driverId'];
3     DatabaseReference driverRef =
4         FirebaseDatabase.instance.ref('drivers/$driverId/location');
5
6     driverRef.onValue.listen((event) {
7         final data = event.snapshot.value as Map<dynamic, dynamic>?;
8         if (data != null) {
9             setState(() {
10                 _driverPosition =
11                     LatLng(data['latitude'].toDouble(), data['longitude'].toDouble());
12                 _updateMapElements();
13             });
14
15             // Di chuyển camera đến vị trí tài xế khi lần đầu load
16             if (_isFirstLoad) {
17                 _animateCameraToDriver();
18                 _isFirstLoad = false;
19             }
20         });
21     });
22 }
23
24 // Di chuyển camera đến vị trí tài xế
25 void _animateCameraToDriver() {
26     if (_driverPosition != const LatLng(0, 0)) {
27         mapController.animateCamera(
28             CameraUpdate.newLatLngZoom(_driverPosition, 15),
29         );
30     }
31 }

```

**Hình 5.6:** Theo dõi vị trí di chuyển của tài xế

### 5.3.3 Kết quả

Như vậy, mỗi khi tài xế di chuyển ít nhất 1m thì vị trí của tài xế sẽ được cập nhật lên Firebase Realtime Database và người dùng sẽ được cập nhật vị trí mới này.

## 5.4 Tính toán chi phí chuyển đi

### 5.4.1 Vấn đề đặt ra

Bên cạnh việc tính toán độ ưu tiên của tài xế, việc tính toán chi phí chuyển đi cũng là một vấn đề cần được giải quyết. Với chi phí hợp lý thì người dùng sẽ cảm thấy hài lòng và sẵn sàng sử dụng dịch vụ.

### 5.4.2 Giải pháp

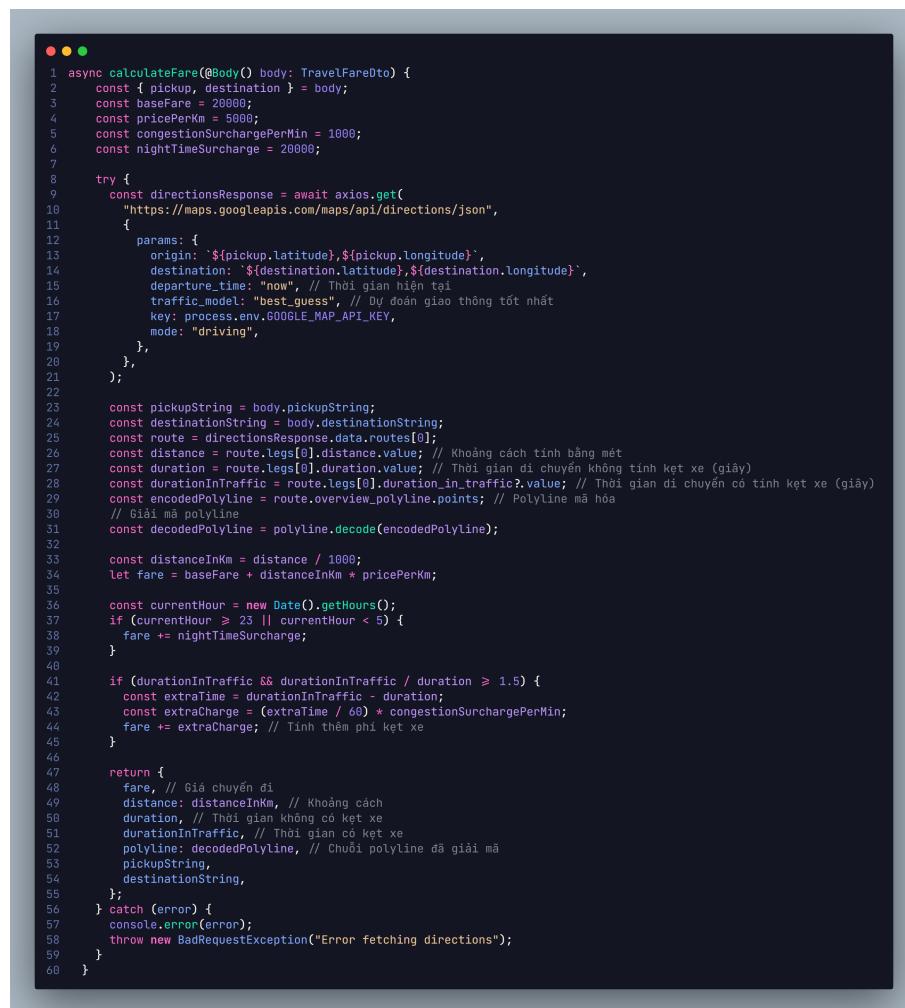
Để tính toán chi phí chuyển đi một cách hợp lý, em đã xây công thức tính toán dựa trên các yếu tố sau:

- Giá cước cơ bản (base fare): 20.000 VNĐ
- Giá theo kilomet: 5.000 VNĐ/km
- Phụ phí kẹt xe: 1.000 VNĐ/phút
- Phụ phí đêm khuya (23h - 5h sáng): 20.000 VNĐ

Quá trình tính toán được thực hiện thông qua các bước sau:

1. Sử dụng Google Maps Directions API để lấy thông tin về:
  - Khoảng cách giữa điểm đón và điểm đến
  - Thời gian di chuyển dự kiến (có và không có kẹt xe)
  - Đường đi chi tiết (polyline)
2. Tính toán chi phí cơ bản dựa trên khoảng cách
3. Kiểm tra thời điểm đặt xe để tính phụ phí đêm khuya
4. Tính toán phụ phí kẹt xe nếu thời gian di chuyển dự kiến vượt quá 50% so với thời gian thông thường

Hình 5.7 minh họa việc tính toán chi phí chuyến đi:



```

1  async calculateFare(@Body() body: TravelFareDto) {
2    const { pickup, destination } = body;
3    const baseFare = 20000;
4    const pricePerKm = 5000;
5    const congestionSurchargePerMin = 1000;
6    const nighttimeSurcharge = 20000;
7
8    try {
9      const directionsResponse = await axios.get(
10        "https://maps.googleapis.com/maps/api/directions/json",
11        {
12          params: {
13            origin: `${pickup.latitude},${pickup.longitude}`,
14            destination: `${destination.latitude},${destination.longitude}`,
15            departure_time: "now", // Thời gian hiện tại
16            traffic_model: "best_guess", // Dự đoán giao thông tốt nhất
17            key: process.env.GOOGLE_MAP_API_KEY,
18            mode: "driving",
19          },
20        },
21      );
22
23      const pickupString = body.pickupString;
24      const destinationString = body.destinationString;
25      const route = directionsResponse.data.routes[0];
26      const distance = route.legs[0].distance.value; // Khoảng cách tính bằng mét
27      const duration = route.legs[0].duration.value; // Thời gian di chuyển không tính kẹt xe (giây)
28      const durationInTraffic = route.legs[0].duration_in_traffic?.value; // Thời gian di chuyển có tính kẹt xe (giây)
29      const encodedPolyline = route.overview_polyline.points; // Polyline mã hóa
30      // Giải mã polyline
31      const decodedPolyline = polyline.decode(encodedPolyline);
32
33      const distanceInKm = distance / 1000;
34      let fare = baseFare + distanceInKm * pricePerKm;
35
36      const currentHour = new Date().getHours();
37      if (currentHour >= 23 || currentHour < 5) {
38        fare += nighttimeSurcharge;
39      }
40
41      if (durationInTraffic && durationInTraffic / duration > 1.5) {
42        const extraTime = durationInTraffic - duration;
43        const extraCharge = (extraTime / 60) * congestionSurchargePerMin;
44        fare += extraCharge; // Tính thêm phí kẹt xe
45      }
46
47      return {
48        fare, // Giá chuyến đi
49        distance: distanceInKm, // Khoảng cách
50        duration, // Thời gian không có kẹt xe
51        durationInTraffic, // Thời gian có kẹt xe
52        polyline: decodedPolyline, // Chuỗi polyline đã giải mã
53        pickupString,
54        destinationString,
55      };
56    } catch (error) {
57      console.error(error);
58      throw new BadRequestException("Error fetching directions");
59    }
60  }

```

Hình 5.7: Tính toán chi phí chuyến đi

### 5.4.3 Kết quả

Thuật toán tính chi phí đã mang lại những kết quả tích cực:

- Chi phí được tính toán minh bạch, dựa trên các yếu tố thực tế

## CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

- Người dùng được thông báo chi phí trước khi bắt đầu chuyến đi
- Phụ phí kẹt xe và đêm khuya được tính hợp lý, phản ánh đúng điều kiện thực tế
- Tính toán được thực hiện tự động và chính xác nhờ tích hợp với Google Maps API

## CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 6.1 Kết luận

Ứng dụng được phát triển trong quá trình thực hiện đồ án đã đáp ứng được các yêu cầu chức năng đã được đặt ra, bao gồm việc tạo một ứng dụng Android để tìm kiếm tài xế và đặt tài xế lái hộ xe. Ứng dụng này giúp người dùng nhanh chóng tìm kiếm tài xế phù hợp và kết nối với họ. Bằng cách áp dụng các nguyên tắc thiết kế phần mềm và phương pháp phân tích thiết kế hệ thống, ứng dụng được xây dựng với khả năng bảo trì dễ dàng và mở rộng linh hoạt trong tương lai. Cụ thể, ứng dụng này được xây dựng dựa trên kiến trúc MVC(Model-View-Controller) để phân tách phần dữ liệu và giao diện giúp cho việc phát triển, bảo trì mã nguồn trở nên dễ dàng hơn.

Bên cạnh đó, việc lựa chọn và sử dụng các công nghệ đã giúp nhanh chóng xây dựng các tính năng và giao diện. Quá trình này đòi hỏi việc nghiên cứu và đánh giá ưu-nhược điểm của từng công nghệ, từ đó chọn ra công nghệ phù hợp với yêu cầu của ứng dụng.

Trong quá trình này, em đã học hỏi và cải thiện được nhiều kỹ năng quan trọng như: quản lý thời gian, tổ chức dự án, nghiên cứu và sử dụng công nghệ mới.

Điều quan trọng nhất mà em rút ra trong thời gian thực hiện đồ án này là tinh thần tìm tòi, học hỏi và việc áp dụng những kiến thức đã được tích lũy trong quá trình học tập vào dự án thực tế. Những kinh nghiệm quý báu này sẽ giúp em áp dụng vào những dự án trong tương lai của em.

### 6.2 Hướng phát triển

Ứng dụng đã đáp ứng được những điều kiện ban đầu tuy nhiên để có thể cạnh tranh được với các ứng dụng đang có trên thị trường, các chức năng sau có thể được thực hiện và phát triển:

- Xây dựng phiên bản để sử dụng trên hệ điều hành IOS.
- Cho phép người dùng có thể chia sẻ vị trí của họ với người khác.
- Tối ưu thuật toán tìm kiếm và phân phối tài xế phù hợp.
- Tối ưu hóa hiển thị và tương tác người dùng.

## MỘT SỐ LUU Ý VỀ TÀI LIỆU THAM KHẢO

Lưu ý: Sinh viên không được đưa bài giảng/slides, các trang Wikipedia, hoặc các trang web thông thường làm tài liệu tham khảo.

Một trang web được phép dùng làm tài liệu tham khảo **chỉ khi** nó là công bố chính thống của cá nhân hoặc tổ chức nào đó. Ví dụ, trang web đặc tả ngôn ngữ XML của tổ chức W3C <https://www.w3.org/TR/2008/REC-xml-20081226/> là TLTK hợp lệ.

Có năm loại tài liệu tham khảo mà sinh viên phải tuân thủ đúng quy định về cách thức liệt kê thông tin như sau. Lưu ý: các phần văn bản trong cặp dấu <> dưới đây chỉ là hướng dẫn khai báo cho từng loại tài liệu tham khảo; sinh viên cần xóa các phần văn bản này trong ĐATN của mình.

**<Bài báo đăng trên tạp chí khoa học:** Tên tác giả, tên bài báo, tên tạp chí, volume, từ trang đến trang (nếu có), nhà xuất bản, năm xuất bản >

**hovy1993automated** E. H. Hovy, "Automated discourse generation using discourse structure relations," *Artificial intelligence*, vol. 63, no. 1-2, pp. 341–385, 1993

**<Sách:** Tên tác giả, tên sách, volume (nếu có), lần tái bản (nếu có), nhà xuất bản, năm xuất bản>

**peterson2007computer** L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

**NguyenThucHai** N. T. Hải, *Mạng máy tính và các hệ thống mở*. Nhà xuất bản giáo dục, 1999.

**<Tập san Báo cáo Hội nghị Khoa học:** Tên tác giả, tên báo cáo, tên hội nghị, ngày (nếu có), địa điểm hội nghị, năm xuất bản>

**poesio2001discourse** M. Poesio and B. Di Eugenio, "Discourse structure and anaphoric accessibility," in *ESSLLI workshop on information structure, discourse structure and discourse semantics*, Copenhagen, Denmark, 2001, pp. 129–143.

**<Đồ án tốt nghiệp, Luận văn Thạc sĩ, Tiến sĩ:** Tên tác giả, tên đồ án/luận văn, loại đồ án/luận văn, tên trường, địa điểm, năm xuất bản>

**knott1996data** A. Knott, "A data-driven methodology for motivating a set of coherence relations," Ph.D. dissertation, The University of Edinburgh, UK, 1996.

**<Tài liệu tham khảo từ Internet:** Tên tác giả (nếu có), tựa đề, cơ quan (nếu

có), địa chỉ trang web, thời gian lần cuối truy cập trang web>

**BernersTim** T. Berners-Lee, *Hypertext transfer protocol (HTTP)*. [Online]. Available: <ftp://info.cern.ch/pub/www/doc/http-spec.txt> (visited on 09/30/2010).

**LectureA** Princeton University, *Wordnet*. [Online]. Available: <http://www.cogsci.princeton.edu/~wn/index.shtml> (visited on 09/30/2010).