

**PHÂN HIỆU TRƯỜNG ĐẠI HỌC THUỶ LỢI**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

****

**ĐỒ ÁN MÔN HỌC**

**THIẾT KẾ & XÂY DỰNG HỆ THỐNG**

**ĐẶT ĐỒ ĂN NHANH (EATNOW)**

|  |  |  |
| --- | --- | --- |
| GVHD | : | TS. |
| SVTH | : | PHAN VĂN VINH |
| MSSV | : | 2251068282 |
| LỚP | : | S25-64CNTT |

**Hồ Chí Minh, Tháng 9 Năm 2025**

Lời mở đầu

Trong bối cảnh xã hội hiện đại, dịch vụ đặt và giao đồ ăn trực tuyến ngày càng trở thành một phần quen thuộc trong đời sống hàng ngày. Các nền tảng giao đồ ăn đã mở ra một kênh kết nối quan trọng giữa khách hàng, nhà hàng và tài xế, mang lại nhiều tiện lợi. Tuy nhiên, đằng sau sự tiện lợi đó, vẫn tồn tại không ít bất cập từ cả ba phía tham gia hệ thống.

* **Khách hàng** thường gặp phải tình trạng thời gian chờ lâu do đơn hàng bị ghép chung nhiều đơn khác, dẫn đến trải nghiệm không như kỳ vọng. Ngoài ra, chi phí phát sinh nhiều phụ phí chồng chéo, cùng với những khó khăn khi thiết lập vị trí giao hàng ở các khu vực hẻm sâu hoặc nơi Google Maps không hiển thị chính xác, khiến họ phải đặt vị trí tạm, dẫn đến sai sót trong giao nhận. Một số ứng dụng còn gặp vấn đề trong trải nghiệm người dùng như thao tác phức tạp, thông báo đơn hàng không ổn định.
* **Nhà hàng** chịu nhiều áp lực từ các nền tảng, khi bị trừ mức chiết khấu cao, thậm chí phải tham gia các chương trình khuyến mãi bắt buộc, gây khó khăn trong việc duy trì lợi nhuận. Bên cạnh đó, việc cập nhật thực đơn hoặc quản lý đơn hàng đôi khi chưa thuận tiện, dẫn đến sai sót và giảm hiệu quả vận hành.
* **Tài xế** là đối tượng trực tiếp chịu ảnh hưởng nặng nề khi thu nhập liên tục bị cắt giảm, trong khi khối lượng công việc ngày càng nhiều. Việc phải xử lý đồng thời nhiều đơn hàng, gặp khó khăn trong tối ưu lộ trình ghép đơn, hay đối diện với sự thiếu minh bạch trong hỗ trợ từ hệ thống khiến trải nghiệm công việc ngày càng áp lực. Ngoài ra, giao diện ứng dụng dành cho tài xế thường chưa được tối ưu, thao tác phức tạp, gây ảnh hưởng đến tốc độ xử lý đơn.

Xuất phát từ những bất cập thực tiễn này, kết hợp với trải nghiệm hơn hai năm trong vai trò tài xế giao đồ ăn, tôi lựa chọn thực hiện đề tài **“EatNow – Hệ thống đặt đồ ăn nhanh”**. Mục tiêu của đề tài là phát triển một hệ thống có khả năng:

* Giúp khách hàng đặt món nhanh chóng, theo dõi đơn hàng trực quan, giảm thiểu sai lệch vị trí.
* Hỗ trợ nhà hàng quản lý thực đơn, đơn hàng và doanh thu một cách minh bạch hơn.
* Cung cấp cho tài xế công cụ thao tác thuận tiện, cập nhật vị trí GPS theo thời gian thực, tối ưu lộ trình và giảm bớt gánh nặng công việc.
* Cho phép quản trị viên theo dõi, điều phối toàn hệ thống, cũng như điều chỉnh chính sách hợp lý.

Đề tài không chỉ là một nghiên cứu học thuật, mà còn mang tính ứng dụng cao, xuất phát từ chính những trải nghiệm thực tế và nhu cầu cấp thiết trong ngành dịch vụ giao đồ ăn. Tôi kỳ vọng hệ thống này có thể phần nào khắc phục các hạn chế hiện nay, đồng thời đem đến trải nghiệm công bằng và hiệu quả hơn cho cả khách hàng, nhà hàng và tài xế.

Mục lục

[CHƯƠNG I: GIỚI THIỆU 2](#_Toc208758588)

[1.1 Giới thiệu đề tài 2](#_Toc208758589)

[1.2 Bối cảnh thị trường và thực trạng 2](#_Toc208758590)

[1.3 Trải nghiệm cá nhân thực tế 3](#_Toc208758591)

[1.4 Những đóng góp của đề tài 4](#_Toc208758592)

[1.5 Phương pháp thực hiện 5](#_Toc208758593)

[1.5.1 Khảo sát và nghiên cứu: 5](#_Toc208758594)

[1.5.2 Phát triển kỹ thuật: 6](#_Toc208758595)

[1.5.3 Kiểm thử và đánh giá: 6](#_Toc208758596)

[1.6 Phạm vi và giới hạn của đề tài 6](#_Toc208758597)

[1.6.1 Phạm vi nghiên cứu: 6](#_Toc208758598)

[1.6.2 Giới hạn về chức năng: 7](#_Toc208758599)

[1.6.3 Giới hạn về công nghệ: 7](#_Toc208758600)

[1.6.4 Giới hạn về quy mô: 7](#_Toc208758601)

[1.6.5 Hướng phát triển tương lai: 7](#_Toc208758602)

[CHƯƠNG II: CƠ SỞ LÝ THUYẾT VÀ PHÂN TÍCH HỆ THỐNG 9](#_Toc208758603)

[2.1 Cơ sở lý thuyết 9](#_Toc208758604)

[2.1.1 Mô hình Client-Server 9](#_Toc208758605)

[2.1.2 Realtime Communication 10](#_Toc208758606)

[2.1.3 Cơ sở dữ liệu 10](#_Toc208758607)

[2.1.4 API và RESTful Design 11](#_Toc208758608)

[2.1.5 Dịch vụ bản đồ và định vị 11](#_Toc208758609)

[2.1.6 Kiến trúc Monolith vs Microservices 12](#_Toc208758610)

[2.2 Khảo sát và phân tích yêu cầu 12](#_Toc208758611)

[2.2.1 Yêu cầu chức năng 12](#_Toc208758612)

[2.2.2 Yêu cầu phi chức năng 14](#_Toc208758613)

[2.2.3 Các tác nhân (Actors) 14](#_Toc208758614)

[2.3 Phân tích hệ thống 15](#_Toc208758615)

[2.3.1 Sơ đồ Use Case 15](#_Toc208758616)

[2.3.2 Sơ đồ hoạt động (Activity Diagram) 15](#_Toc208758617)

[2.3.3 Sơ đồ ERD (Entity Relationship Diagram) 15](#_Toc208758618)

[2.3.4 Mô hình tổng quan hệ thống 15](#_Toc208758619)

[2.4 Kết luận chương 2 15](#_Toc208758620)

CHƯƠNG I: GIỚI THIỆU

1.1 Giới thiệu đề tài

EatNow — Hệ thống đặt đồ ăn nhanh là giải pháp prototype kết nối khách hàng, nhà hàng và tài xế theo quy trình đặt – xử lý – giao – nhận. Đề tài hướng tới cải thiện trải nghiệm người dùng, giảm thiểu thời gian đặt và giao, tối ưu thu nhập cho tài xế và đảm bảo minh bạch chi phí cho nhà hàng. Ngoài nghiên cứu lý thuyết, tác giả còn dựa trên trải nghiệm thực tế hơn hai năm làm tài xế giao đồ ăn, từ đó xác định rõ những khó khăn thường gặp và đề xuất giải pháp khắc phục.

Mục tiêu chính của đề tài:

* Xây dựng hệ thống đặt đồ ăn với kiến trúc hiện đại, hỗ trợ real-time communication
* Tối ưu hóa trải nghiệm người dùng thông qua giao diện đơn giản và thân thiện
* Giải quyết các vấn đề thực tế trong ngành giao đồ ăn dựa trên kinh nghiệm cá nhân
* Ứng dụng các công nghệ web và mobile hiện đại (NestJS, Next.js, Flutter, Socket.IO)

1.2 Bối cảnh thị trường và thực trạng

Ngành giao đồ ăn trực tuyến tại Việt Nam tăng trưởng mạnh mẽ trong những năm gần đây, với quy mô dự kiến đạt hàng tỷ USD vào năm 2025 (Statista, 2024). Thị trường hiện bị chi phối bởi hai nền tảng lớn: ShopeeFood và GrabFood, chiếm lần lượt khoảng 56% và 41% thị phần tại các thành phố lớn (Q&Me, 2024).

Những thành công của các nền tảng hiện tại:

* Xây dựng mạng lưới đối tác rộng (nhà hàng, tài xế, khách hàng)
* Cung cấp hạ tầng kỹ thuật đủ mạnh để xử lý lượng lớn đơn hàng
* Đa dạng phương thức thanh toán, nhiều chương trình khuyến mãi thu hút người dùng
* Tích hợp công nghệ AI để tối ưu routing và dự đoán thời gian giao hàng

Những hạn chế và thách thức:Đối với khách hàng:

* Thời gian giao hàng bị kéo dài do ghép đơn, vị trí giao khó thiết lập chính xác ở khu vực hẻm sâu (Nguyen, 2023)
* Phí giao hàng và dịch vụ tăng cao, đặc biệt trong giờ cao điểm
* Giao diện ứng dụng phức tạp với nhiều bước thao tác không cần thiết
* Thông báo đẩy (push notification) không ổn định, dẫn đến bỏ lỡ đơn hàng

Đối với nhà hàng:

* Mức chiết khấu cao (từ 25–30%), nhiều nơi không có lợi nhuận do bắt buộc tham gia các chương trình khuyến mãi (VnExpress, 2024)
* Phụ thuộc vào thuật toán của nền tảng để hiển thị trong kết quả tìm kiếm
* Khó khăn trong việc quản lý đơn hàng khi có nhiều nền tảng cùng lúc
* Thiếu công cụ phân tích dữ liệu khách hàng hiệu quả

Đối với tài xế:

* Thu nhập giảm dần do bị cắt thưởng, tính khoảng cách giao hàng đôi khi không minh bạch, thủ tục khiếu nại rườm rà (Pham, 2023; Tien Phong, 2024)
* Áp lực hoàn thành đơn hàng trong thời gian ngắn, ảnh hưởng đến an toàn giao thông
* Hệ thống định vị không chính xác ở một số khu vực, đặc biệt là hẻm nhỏ
* Thiếu hỗ trợ kỹ thuật khi gặp sự cố trong quá trình giao hàng

Về mặt kỹ thuật:

* Ứng dụng gặp vấn đề về thông báo đẩy (push notification không ổn định)
* Giao diện còn nhiều bước thao tác, chưa tối ưu trải nghiệm
* Hệ thống đôi khi bị lag hoặc crash trong giờ cao điểm
* Thiếu tính năng real-time tracking chính xác cho khách hàng

1.3 Trải nghiệm cá nhân thực tế

Trong hơn hai năm làm tài xế giao đồ ăn, tác giả ghi nhận nhiều tình huống điển hình phản ánh các hạn chế trên. Ví dụ:**Vấn đề về ghép đơn và routing:**"Có lần tôi nhận cùng lúc ba đơn từ ba quán khác nhau. Hệ thống ghép đơn nhưng lộ trình đi vòng vèo, khiến khách hàng cuối cùng phải chờ hơn 40 phút. Khi liên hệ hỗ trợ để hủy một đơn, tôi mất gần 10 phút chờ tổng đài." (Phan, ghi chú cá nhân, 2024).**Vấn đề về định vị và địa chỉ:**"Tại một số hẻm nhỏ, Google Maps không nhận diện được vị trí. Khách hàng thường đặt nhầm địa chỉ gần đó, dẫn đến mất thời gian gọi điện và tìm đường." (Phan, ghi chú cá nhân, 2024).**Vấn đề về thu nhập và minh bạch:**"Thu nhập của tôi giảm từ 800.000 VNĐ/ngày xuống còn 500.000 VNĐ/ngày trong vòng 6 tháng, mặc dù số đơn hàng không thay đổi. Hệ thống không giải thích rõ cách tính phí và thưởng." (Phan, ghi chú cá nhân, 2024).**Vấn đề về hỗ trợ khách hàng:**"Khi khách hàng khiếu nại về chất lượng đồ ăn hoặc thời gian giao hàng, tôi thường phải chịu trách nhiệm mặc dù không phải lỗi của mình. Quy trình giải quyết khiếu nại rất phức tạp và mất thời gian." (Phan, ghi chú cá nhân, 2024).Những trải nghiệm này khớp với các vấn đề được phản ánh trong nhiều nghiên cứu và bài báo, cho thấy đây không phải hiện tượng cá biệt mà là hạn chế mang tính hệ thống.

1.4 Những đóng góp của đề tài

Đề tài EatNow hướng đến giải quyết các vấn đề thực tế thông qua các đóng góp sau:

1. Thiết kế hệ thống realtime cho tài xế nhận đơn tức thì:

* + Sử dụng WebSocket (Socket.IO) để đảm bảo thông báo đơn hàng đến tài xế ngay lập tức
  + Tối ưu hóa thuật toán phân phối đơn hàng dựa trên vị trí và tình trạng sẵn sàng của tài xế

2. Hỗ trợ nhập vị trí chi tiết để giảm sai lệch bản đồ:

* + Cho phép khách hàng thêm ghi chú chi tiết về địa chỉ giao hàng
  + Tích hợp tính năng chụp ảnh địa điểm để tài xế dễ nhận diện
  + Hỗ trợ định vị GPS chính xác với Flutter Geolocator

3. Minh bạch hóa phí và hoa hồng với nhà hàng:

* + Hiển thị rõ ràng các loại phí và tỷ lệ hoa hồng
  + Cung cấp dashboard thống kê chi tiết về doanh thu và chi phí
  + Tính toán lợi nhuận thực tế sau khi trừ các khoản phí

4. Tối giản giao diện tài xế để rút ngắn thao tác:

* + Thiết kế UI/UX đơn giản, tập trung vào các chức năng cốt lõi
  + Giảm số bước thao tác từ nhận đơn đến hoàn thành giao hàng
  + Tích hợp voice commands và shortcuts cho các thao tác thường dùng

5. Cung cấp dashboard thống kê cơ bản cho quản trị viên và nhà hàng:

* + Dashboard real-time cho admin theo dõi hoạt động hệ thống
  + Báo cáo doanh thu và phân tích xu hướng cho nhà hàng
  + Thống kê hiệu suất và thu nhập cho tài xế

6. Ứng dụng công nghệ hiện đại:

* + Kiến trúc microservices-ready với NestJS backend
  + Frontend responsive với Next.js và Tailwind CSS
  + Mobile app cross-platform với Flutter
  + Database NoSQL (MongoDB) cho scalability và flexibility

1.5 Phương pháp thực hiện

1.5.1 Khảo sát và nghiên cứu:

* **Khảo sát thực tế:** Từ trải nghiệm tác giả và phỏng vấn nhanh các tài xế, nhà hàng, khách hàng
* **Nghiên cứu thứ cấp:** Báo cáo thị trường, báo chí, tài liệu nghiên cứu về ngành giao đồ ăn
* **Phân tích đối thủ:** Nghiên cứu các tính năng và hạn chế của ShopeeFood, GrabFood

1.5.2 Phát triển kỹ thuật:

* **Backend:** NestJS với TypeScript, RESTful API, Socket.IO cho real-time
* **Frontend:** Next.js 14 với React 18, Tailwind CSS, Zustand state management
* **Mobile:** Flutter với Dart, Firebase integration
* **Database:** MongoDB Atlas, Redis cache, PostgreSQL backup
* **Authentication:** JWT tokens với role-based access control
* **Deployment:** Render (backend), Vercel (frontend), Firebase (mobile)

1.5.3 Kiểm thử và đánh giá:

* **Unit testing:** Kiểm thử từng module và service
* **Integration testing:** Kiểm thử luồng đặt – giao hàng qua kịch bản mô phỏng
* **User acceptance testing:** Đánh giá trải nghiệm người dùng với các actor khác nhau
* **Performance testing:** Kiểm tra khả năng xử lý concurrent users và response time

1.6 Phạm vi và giới hạn của đề tài

1.6.1 Phạm vi nghiên cứu:

Mặc dù các nền tảng lớn như ShopeeFood hay GrabFood đã xây dựng hệ sinh thái hoàn chỉnh với nhiều tính năng hiện đại, đề tài EatNow không nhằm so sánh trực tiếp hay thay thế các hệ thống thương mại này. Thay vào đó, dự án được thực hiện với mục đích:

* Thử nghiệm mô hình đặt – giao đồ ăn theo cách tiếp cận đơn giản, phù hợp với phạm vi của một đồ án sinh viên
* Học hỏi từ những thành công và hạn chế của các nền tảng hiện có để áp dụng vào thiết kế thử nghiệm
* Rèn luyện kỹ năng phân tích, thiết kế, lập trình hệ thống phân tán có yếu tố realtime

1.6.2 Giới hạn về chức năng:

Do thời gian và nguồn lực có hạn, hệ thống chỉ tập trung vào các chức năng cốt lõi:

* Đặt món và quản lý giỏ hàng
* Nhận đơn và cập nhật trạng thái real-time
* Theo dõi vị trí tài xế và trạng thái giao hàng
* Bản đồ lộ trình cơ bản
* Quản lý người dùng và phân quyền
* Dashboard thống kê cơ bản

1.6.3 Giới hạn về công nghệ:

* Chưa tích hợp thanh toán online (chỉ hỗ trợ COD và chuyển khoản)
* Chưa có hệ thống đánh giá và review chi tiết
* Chưa tích hợp AI/ML cho recommendation và optimization
* Chưa có hệ thống loyalty program và marketing automation
* Chưa hỗ trợ multi-language và internationalization

1.6.4 Giới hạn về quy mô:

* Prototype chỉ hỗ trợ một thành phố/region
* Chưa có hệ thống logistics phức tạp cho nhiều warehouse
* Chưa tích hợp với các hệ thống ERP của nhà hàng
* Chưa có hệ thống customer service và support ticket

1.6.5 Hướng phát triển tương lai:

* Mở rộng sang các thành phố khác
* Tích hợp thanh toán online và ví điện tử
* Phát triển AI recommendation system
* Xây dựng hệ thống logistics và inventory management
* Triển khai microservices architecture cho scalability

CHƯƠNG II: CƠ SỞ LÝ THUYẾT VÀ PHÂN TÍCH HỆ THỐNG

2.1 Cơ sở lý thuyết

Sau khi trải nghiệm thực tế làm tài xế giao đồ ăn, mình nhận ra rằng việc xây dựng một hệ thống đặt đồ ăn không chỉ đơn thuần là lập trình mà còn cần hiểu sâu về cách các thành phần tương tác với nhau. Chương này sẽ trình bày những kiến thức nền tảng mà mình đã học được trong quá trình nghiên cứu và phát triển EatNow.

2.1.1 Mô hình Client-Server

Khi bắt đầu thiết kế EatNow, mình phải đối mặt với câu hỏi: làm sao để khách hàng, nhà hàng, tài xế và admin có thể tương tác với nhau một cách mượt mà? Câu trả lời nằm ở kiến trúc Client-Server - một mô hình đã được chứng minh qua nhiều năm phát triển ứng dụng web.**Backend (Server) - Trái tim của hệ thống:**Mình chọn NestJS làm framework chính vì nó mang lại cảm giác "quen thuộc" như Spring Boot mà mình đã học, nhưng lại sử dụng TypeScript - ngôn ngữ mà mình cảm thấy an toàn hơn JavaScript thuần. NestJS cung cấp:

* **Kiến trúc modular**: Mỗi module như auth, order, restaurant có thể phát triển độc lập
* **Decorators**: Giúp code trở nên sạch sẽ và dễ đọc hơn
* **Dependency Injection**: Quản lý dependencies một cách tự động

Express.js được tích hợp sẵn trong NestJS, đảm bảo hiệu năng cao cho RESTful API. Socket.IO được chọn cho real-time communication vì mình từng gặp vấn đề với thông báo đẩy không ổn định khi làm tài xế - điều này sẽ được giải quyết bằng WebSocket.**Frontend (Client) - Giao diện người dùng:**Next.js 14 được chọn vì mình muốn có một ứng dụng web hiện đại với Server-Side Rendering (SSR) để tối ưu SEO và tốc độ tải trang. React 18 với hooks giúp quản lý state dễ dàng hơn, trong khi Tailwind CSS cho phép mình thiết kế giao diện nhanh chóng mà không cần viết CSS thuần.Zustand được chọn thay vì Redux vì nó đơn giản hơn và phù hợp với quy mô prototype. React Query giúp mình quản lý data fetching và caching một cách hiệu quả.**Mobile (Client) - Trải nghiệm di động:**Flutter được chọn cho mobile app vì mình muốn có một ứng dụng cross-platform mà không cần maintain hai codebase riêng biệt cho iOS và Android. Geolocator package sẽ giải quyết vấn đề định vị GPS mà mình từng gặp phải khi làm tài xế.

2.1.2 Realtime Communication

Đây là phần mà mình cảm thấy quan trọng nhất, vì trong thực tế, việc thông báo đơn hàng đến tài xế chậm trễ có thể dẫn đến khách hàng hủy đơn. Khi làm tài xế, mình thường phải chờ 5-10 phút mới nhận được thông báo đơn hàng mới, trong khi khách hàng đã đặt từ lâu.**WebSocket với Socket.IO:**Socket.IO được chọn vì nó có khả năng fallback về polling khi WebSocket không khả dụng - điều này rất quan trọng khi tài xế đang di chuyển và mạng không ổn định.**Cơ chế Rooms:**Mình thiết kế hệ thống rooms để mỗi actor có thể nhận thông báo riêng:

* customer\_${id}: Khách hàng theo dõi trạng thái đơn hàng
* restaurant\_${id}: Nhà hàng nhận thông báo đơn mới
* driver\_${id}: Tài xế nhận đơn và cập nhật vị trí

Events quan trọng:

* new\_order: Thông báo đơn hàng mới đến nhà hàng
* order\_update: Cập nhật trạng thái đơn hàng cho tất cả parties
* driver\_location\_update: Theo dõi vị trí tài xế real-time

2.1.3 Cơ sở dữ liệu

Khi thiết kế database, mình phải cân nhắc giữa SQL và NoSQL. Sau khi nghiên cứu và thử nghiệm, mình quyết định sử dụng MongoDB làm database chính vì:

MongoDB (NoSQL) - Database chính:

* **Schema linh hoạt**: Điều này rất quan trọng khi mình chưa chắc chắn về cấu trúc dữ liệu cuối cùng
* **JSON-native**: Phù hợp với JavaScript/TypeScript ecosystem
* **Horizontal scaling**: Dễ dàng mở rộng khi hệ thống phát triển

Mongoose ODM giúp mình validate dữ liệu và tạo relationships giữa các collections một cách dễ dàng.**Redis (Cache & Session):**Redis được sử dụng để cache dữ liệu thường xuyên truy cập và quản lý session. Khi làm tài xế, mình nhận thấy việc load lại thông tin profile và lịch sử đơn hàng rất chậm - Redis sẽ giải quyết vấn đề này.**PostgreSQL (Relational - Backup):**Mình vẫn giữ PostgreSQL như một backup option, phòng trường hợp cần structured data cho reporting và analytics.

2.1.4 API và RESTful Design

Thiết kế API là một thách thức lớn vì mình phải đảm bảo tính nhất quán và dễ sử dụng. Mình áp dụng RESTful principles:**RESTful API Design:**

* **Endpoints rõ ràng**: /auth, /users, /restaurants, /orders
* **HTTP methods chuẩn**: GET cho đọc, POST cho tạo mới, PUT/PATCH cho cập nhật, DELETE cho xóa
* **Status codes nhất quán**: 200 cho thành công, 400 cho lỗi client, 500 cho lỗi server

**Authentication & Authorization:**JWT được chọn vì tính stateless và khả năng scale. Mình thiết kế hệ thống phân quyền với 4 roles: customer, restaurant, driver, admin. Mỗi role có quyền truy cập khác nhau vào các endpoints.

2.1.5 Dịch vụ bản đồ và định vị

Đây là phần mà mình có nhiều kinh nghiệm thực tế nhất. Khi làm tài xế, mình thường gặp vấn đề với:

* Google Maps không nhận diện được địa chỉ ở hẻm nhỏ
* Tính toán khoảng cách không chính xác
* Routing không tối ưu

**Flutter Geolocator:**Package này giúp mình lấy vị trí GPS chính xác và theo dõi movement của tài xế real-time. Mình cũng tích hợp permission handling để đảm bảo app hoạt động mượt mà.**Map Services Integration:**Mình chuẩn bị tích hợp cả Google Maps API và OpenStreetMap để có backup option khi một service không khả dụng.

2.1.6 Kiến trúc Monolith vs Microservices

Đây là một quyết định quan trọng mà mình phải cân nhắc kỹ. Ban đầu mình nghĩ đến microservices vì nó "cool" và hiện đại, nhưng sau khi cân nhắc:**Lý do chọn Monolith:**

* **Development speed**: Mình muốn hoàn thành prototype nhanh để test ý tưởng
* **Debugging dễ dàng**: Khi có lỗi, mình chỉ cần trace trong một codebase
* **Deployment đơn giản**: Chỉ cần deploy một service thay vì nhiều services
* **Shared database transactions**: Đảm bảo data consistency

Nhược điểm mình chấp nhận:

* **Scaling limitations:** Nhưng với quy mô prototype thì không sao
* **Technology lock-in:** Nhưng mình có thể refactor sau
* **Single point of failure**: Có thể giải quyết bằng load balancer

**Tương lai Microservices:**Mình đã thiết kế code theo modular pattern để dễ dàng tách thành microservices sau này khi hệ thống phát triển.

2.2 Khảo sát và phân tích yêu cầu

Sau khi trải nghiệm thực tế và nghiên cứu các nền tảng hiện có, mình đã xác định được những yêu cầu cụ thể cho EatNow. Phần này sẽ trình bày chi tiết về những gì hệ thống cần đáp ứng.

2.2.1 Yêu cầu chức năng

**Khách hàng (Customer) - Trải nghiệm đặt hàng mượt mà:**Dựa trên kinh nghiệm của mình khi quan sát khách hàng trong quá trình giao hàng, mình nhận thấy họ cần:

* **Đăng ký/đăng nhập đơn giản**: Không muốn quá nhiều bước xác thực
* **Duyệt nhà hàng trực quan**: Giao diện đẹp với hình ảnh món ăn rõ ràng
* **Giỏ hàng linh hoạt**: Có thể thêm/bớt món, lưu lại cho lần sau
* **Theo dõi đơn hàng real-time**: Biết được tài xế đang ở đâu, bao giờ đến
* **Thanh toán đa dạng**: COD, chuyển khoản, ví điện tử

**Nhà hàng (Restaurant) - Quản lý hiệu quả:**Từ những cuộc trò chuyện với chủ nhà hàng, mình hiểu được họ cần:

* **Quản lý menu dễ dàng**: Thêm/sửa/xóa món ăn nhanh chóng
* **Nhận đơn hàng tức thì**: Thông báo real-time khi có đơn mới
* **Cập nhật trạng thái đơn**: Từ "đang chuẩn bị" đến "sẵn sàng giao"
* **Thống kê doanh thu**: Biết được thu nhập hàng ngày, tuần, tháng
* **Quản lý khuyến mãi**: Tạo và quản lý các chương trình giảm giá

**Tài xế (Driver) - Công việc thuận lợi:**Dựa trên kinh nghiệm cá nhân, mình thiết kế các tính năng để giải quyết những khó khăn mà tài xế thường gặp:

* **Nhận đơn hàng nhanh**: Thông báo tức thì, không bị miss đơn
* **Cập nhật vị trí GPS**: Khách hàng biết được tài xế đang ở đâu
* **Cập nhật trạng thái giao hàng**: "Đã lấy đồ", "Đang giao", "Đã giao"
* **Xem thu nhập chi tiết**: Biết được mỗi đơn kiếm được bao nhiêu
* **Giao diện đơn giản**: Ít thao tác, tập trung vào việc giao hàng

**Quản trị viên (Admin) - Kiểm soát toàn hệ thống:**Admin cần có cái nhìn tổng quan về toàn bộ hoạt động:

* **Quản lý người dùng**: Duyệt đăng ký nhà hàng/tài xế mới
* **Theo dõi hoạt động**: Số đơn hàng, doanh thu, tài xế online
* **Xử lý khiếu nại**: Giải quyết các vấn đề phát sinh
* **Thống kê tổng quan**: Báo cáo hiệu suất hệ thống

2.2.2 Yêu cầu phi chức năng

**Hiệu năng - Đảm bảo trải nghiệm mượt mà:**Mình đặt mục tiêu hệ thống có thể hỗ trợ 1000+ concurrent users vì trong giờ cao điểm, số lượng đơn hàng có thể tăng đột biến. Response time phải dưới 2 giây để không làm người dùng chờ đợi.**Bảo mật - Bảo vệ thông tin người dùng:**JWT authentication được chọn vì tính stateless và khả năng scale. Mình cũng implement role-based authorization để đảm bảo mỗi user chỉ có thể truy cập đúng chức năng của mình.**Khả năng mở rộng - Chuẩn bị cho tương lai:**Mặc dù hiện tại là monolith, mình đã thiết kế code theo modular pattern để dễ dàng tách thành microservices khi cần scale.**Độ tin cậy - Đảm bảo hệ thống ổn định:**Mình đặt mục tiêu 99.9% uptime và implement error handling toàn diện để xử lý các tình huống bất thường.

2.2.3 Các tác nhân (Actors)

Primary Actors:

* **Customer**: Người đặt đồ ăn - đây là đối tượng mà mình muốn mang lại trải nghiệm tốt nhất
* **Restaurant**: Nhà hàng cung cấp dịch vụ - đối tác quan trọng của hệ thống
* **Driver**: Tài xế giao hàng - những người mà mình hiểu rõ nhất về nhu cầu và khó khăn
* **Admin**: Quản trị viên hệ thống - người kiểm soát toàn bộ hoạt động

Secondary Actors:

* **Payment Gateway**: Xử lý thanh toán (mình chưa tích hợp trong prototype này)
* **Map Service**: Cung cấp dịch vụ bản đồ và routing
* **Notification Service**: Gửi thông báo push đến mobile app

2.3 Phân tích hệ thống

2.3.1 Sơ đồ Use Case

.

2.3.2 Sơ đồ hoạt động (Activity Diagram)

2.3.3 Sơ đồ ERD (Entity Relationship Diagram)

Database được thiết kế với 5 entities chính: User, Restaurant, Item, Order, Driver. Mình đã cân nhắc kỹ về relationships và constraints để đảm bảo data integrity.

2.3.4 Mô hình tổng quan hệ thống

Kiến trúc tổng thể được thiết kế với 3 layers chính: Client (Frontend/Mobile), Backend API, và Database Layer. Mình cũng chuẩn bị tích hợp các external services như Maps API và Payment Gateway.

2.4 Kết luận chương 2

Chương này đã trình bày cơ sở lý thuyết và phân tích hệ thống cho EatNow. Với mình, những phân tích này không chỉ để làm báo cáo, mà còn giúp mình nhìn rõ hơn về cách một ứng dụng delivery thực sự vận hành ở backend - từ database đến real-time socket.**Những điều mình học được:**

* Thiết kế hệ thống không chỉ là chọn công nghệ "hot" nhất, mà phải phù hợp với quy mô và mục tiêu dự án
* Realtime communication là yếu tố quan trọng quyết định trải nghiệm người dùng
* Database design cần cân nhắc cả hiện tại và tương lai
* API design phải nhất quán và dễ sử dụng

Những thách thức mình gặp phải:

* Cân bằng giữa tính năng và độ phức tạp
* Thiết kế database schema linh hoạt nhưng vẫn đảm bảo performance
* Chọn công nghệ phù hợp với timeline và resources

**Hướng phát triển:**Mặc dù hiện tại là prototype, mình đã thiết kế hệ thống với khả năng mở rộng trong tương lai. Khi có điều kiện, mình sẽ refactor thành microservices và tích hợp thêm các tính năng như AI recommendation và advanced analytics.Những phân tích này tạo nền tảng vững chắc cho việc thiết kế và triển khai hệ thống trong chương tiếp theo, đảm bảo EatNow không chỉ là một bài tập học thuật mà còn là một sản phẩm thực tế có thể giải quyết những vấn đề mà mình đã trải nghiệm trong quá trình làm tài xế giao đồ ăn.

CHƯƠNG III:

### Luồng đăng nhập end-to-end (cookie-based)

- Bắt đầu (Client)

- Người dùng nhập email + password trên trang login (Customer/Restaurant).

- FE gọi API:

- POST `${API}/api/v1/auth/login`

- headers: `Content-Type: application/json`

- body: `{ email, password }`

- fetch luôn kèm `credentials: 'include'` để gửi/nhận cookie.

- Xác thực (Backend)

- `AuthController.login` → `AuthService.validateUser` kiểm tra email/password (bcrypt).

- Nếu hợp lệ:

- Tạo `access\_token` (JWT) → Set cookie `access\_token` (HttpOnly, SameSite=Lax, Path=/).

- Trả về JSON: `{ access\_token, user: { id, email, role, ... } }`

- access\_token cũng có trong payload (cho debug/FE), nhưng FE không lưu; cookie mới là nguồn thật.

- Logic bổ sung:

- Nếu role `customer`: auto-create `Customer` profile nếu chưa có và link `users.customerProfile`.

- Nếu role `restaurant`: đảm bảo có `Restaurant` của owner và link `users.restaurantProfile` (đã chỉnh).

- Lưu trữ (Client)

- FE KHÔNG lưu token vào `localStorage/sessionStorage`.

- FE chỉ giữ state trong bộ nhớ (context/recoil/state) cho UI, ví dụ `user` để hiển thị tên/role.

- Mọi request sau login đều `credentials: 'include'` để cookie tự đính kèm.

- Kiểm tra phiên (Client khởi tạo)

- Ngay khi load app, `AuthContext` gọi:

- GET `${API}/api/v1/auth/me` (credentials: 'include')

- BE đọc cookie `access\_token` → verify → trả `{ id, email, role, name }`.

- FE set `isAuthenticated=true`, `user=...` trong context.

- Phân quyền trang nhà hàng

- `restaurant/layout.tsx`:

- Nếu chưa xác thực → redirect `/restaurant/login`.

- Nếu đã xác thực:

- Gọi GET `${API}/api/v1/restaurants/mine` (credentials: include)

- BE dùng cookie → biết user id → trả về restaurant của chủ.

- FE render sidebar/header, và lưu `restaurant` trong state để các trang con dùng.

- Tương tác tiếp theo (ví dụ Thêm món)

- FE tìm `restaurantId` bằng cách gọi `resolveRestaurant()`:

- Ưu tiên `/restaurants/mine` (cookie-based) → lấy đúng `\_id`.

- Gọi POST `${API}/api/v1/restaurants/:restaurantId/items`

- body: `{ name, price, type, description?, imageUrl?, categoryId?, isActive?, position? }`

- credentials: include

- BE guard/ownership:

- `JwtAuthGuard` + (OwnershipGuard ở endpoints yêu cầu) → xác minh user là owner của `:restaurantId`.

- Trả về item mới → FE refresh danh sách.

- Trang người dùng (customer)

- FE dùng các endpoint public:

- GET `/api/v1/restaurants/public` (danh sách)

- GET `/api/v1/restaurants/categories/public` (global categories)

- GET `/api/v1/restaurants/public/by-category?category=...`

- Khi cần dữ liệu cá nhân (đơn hàng): dùng endpoints có guard, gửi cookie.

- Logout

- FE gọi POST `${API}/api/v1/auth/logout` (credentials: include).

- BE xóa cookie `access\_token` (+ các role cookies nếu có).

- FE xóa state UI (user/restaurant trong context) và điều hướng về trang login.

### Dữ liệu đi/đến (tóm tắt)

- Request login (FE→BE): `{ email, password }`

- Response login (BE→FE): set-cookie `access\_token`; body `{ access\_token, user }` (FE không lưu token).

- Xác thực về sau: tất cả request gửi cookie tự động nhờ `credentials: 'include'`.

- FE lưu gì:

- `user` (state context) để hiển thị.

- `restaurant` (state context/layout) cho khu vực nhà hàng.

- Không lưu token vào localStorage; đã gỡ các phụ thuộc `eatnow\_restaurant\_id`.

### Lưu ý bảo mật/ổn định

- CORS bật `credentials: true`, `origin` whitelist theo env.

- Cookie HttpOnly giảm nguy cơ XSS đánh cắp token.

- Không auto-create restaurant trong GET `/restaurants/mine` (đã tắt) để tránh trùng dữ liệu.

- Nếu DB bị clear: login lại → `/restaurants/mine` sẽ trả null; sau khi đăng ký role restaurant hoặc seed, luồng hoạt động bình thường.

**�� TỔNG QUAN HOÀN CHỈNH CÁC BẢNG VÀ THUỘC TÍNH**

**��️ CẤU TRÚC CƠ BẢN (9 bảng)**

**1. User (Bảng người dùng)**

* **Tên bảng**: users
* **Mục đích**: Quản lý thông tin người dùng hệ thống
* **Thuộc tính**:
* email (String, unique, required) - Email đăng nhập
* password (String, required) - Mật khẩu
* name (String, required) - Tên người dùng
* phone (String, optional) - Số điện thoại
* role (Enum, required) - Vai trò: admin, customer, driver, restaurant
* avatarUrl (String, optional) - URL ảnh đại diện
* addresses (Array, optional) - Danh sách địa chỉ:
* label (String) - Nhãn địa chỉ
* addressLine (String) - Địa chỉ chi tiết
* latitude (Number) - Vĩ độ
* longitude (Number) - Kinh độ
* note (String) - Ghi chú
* isDefault (Boolean) - Địa chỉ mặc định
* addressLabels (Array[String], default: ['Nhà', 'Chỗ làm', 'Nhà mẹ chồng']) - Danh sách nhãn mặc định

**2. Driver (Bảng tài xế) - ✅ REFACTOR HOÀN TOÀN**

* **Tên bảng**: drivers
* **Mục đích**: Quản lý thông tin tài xế giao hàng độc lập
* **Thuộc tính**:
* userId (ObjectId, ref: User, required) - ID người dùng
* name (String, required) - Tên tài xế
* phone (String, required) - Số điện thoại
* email (String, required) - Email
* status (Enum, default: 'inactive') - Trạng thái: inactive, active, suspended, offline
* locationType (String, default: 'Point') - Loại vị trí
* location (Array[Number], 2dsphere index) - Tọa độ [lng, lat]
* lastLocationAt (Date, optional) - Thời gian cập nhật vị trí cuối
* totalDeliveries (Number, default: 0) - Tổng số đơn giao
* rating (Number, default: 0) - Đánh giá trung bình
* ratingCount (Number, default: 0) - Số lượng đánh giá
* licenseNumber (String, optional) - Số bằng lái
* vehicleType (String, optional) - Loại xe: motorbike, car, bicycle
* vehicleNumber (String, optional) - Biển số xe
* isAvailable (Boolean, default: true) - Có sẵn sàng nhận đơn
* currentOrderId (ObjectId, optional) - ID đơn hàng hiện tại

**3. Restaurant (Bảng nhà hàng)**

* **Tên bảng**: restaurants
* **Mục đích**: Quản lý thông tin nhà hàng
* **Thuộc tính**:
* ownerUserId (ObjectId, ref: User) - ID chủ nhà hàng
* name (String, required) - Tên nhà hàng
* status (String, default: 'pending') - Trạng thái: pending, active, suspended, closed
* joinedAt (Date, default: Date.now) - Ngày tham gia
* followersCount (Number, default: 0) - Số lượng người theo dõi
* description (String, optional) - Mô tả
* address (String, optional) - Địa chỉ
* openingHours (String, optional) - Giờ mở cửa
* openTime (String, optional) - Giờ mở (HH:mm)
* closeTime (String, optional) - Giờ đóng (HH:mm)
* openDays (Array[Number], optional) - Các ngày mở cửa (0-6)
* latitude (Number, optional) - Vĩ độ
* longitude (Number, optional) - Kinh độ

**4. Category (Bảng danh mục món ăn)**

* **Tên bảng**: categories
* **Mục đích**: Phân loại món ăn trong nhà hàng
* **Thuộc tính**:
* restaurantId (ObjectId, ref: Restaurant, required) - ID nhà hàng
* name (String, required) - Tên danh mục
* position (Number, default: 0) - Vị trí sắp xếp

**5. Item (Bảng món ăn)**

* **Tên bảng**: items
* **Mục đích**: Quản lý thông tin món ăn
* **Thuộc tính**:
* restaurantId (ObjectId, ref: Restaurant, required) - ID nhà hàng
* categoryId (ObjectId, ref: Category, optional) - ID danh mục
* name (String, required) - Tên món ăn
* nameSearch (String, optional) - Tên tìm kiếm (không dấu)
* price (Number, required) - Giá
* type (Enum, required) - Loại: food, drink
* description (String, optional) - Mô tả
* imageUrl (String, optional) - URL ảnh
* imageId (ObjectId, ref: Image, optional) - ID ảnh
* rating (Number, default: 0, min: 0, max: 5) - Đánh giá
* reviewCount (Number, default: 0) - Số lượng đánh giá
* isActive (Boolean, default: true) - Trạng thái hoạt động
* position (Number, default: 0) - Vị trí sắp xếp
* popularityScore (Number, default: 0) - Điểm phổ biến

**🛒 CẤU TRÚC ĐƠN HÀNG (2 bảng)**

**6. Order (Bảng đơn hàng) - ✅ CẢI THIỆN**

* **Tên bảng**: orders
* **Mục đích**: Quản lý đơn hàng
* **Thuộc tính**:
* customerId (ObjectId, ref: User, required) - ID khách hàng
* restaurantId (ObjectId, ref: Restaurant, required) - ID nhà hàng
* items (Array[OrderItem], required) - Danh sách món ăn
* total (Number, required) - Tổng tiền món ăn
* deliveryFee (Number, required) - Phí giao hàng
* finalTotal (Number, required) - Tổng tiền cuối cùng
* deliveryAddress (Object, required) - **CẢI THIỆN**: Địa chỉ giao hàng có tọa độ:
* label (String) - Nhãn địa chỉ
* addressLine (String) - Địa chỉ chi tiết
* latitude (Number) - Vĩ độ
* longitude (Number) - Kinh độ
* note (String, optional) - Ghi chú
* specialInstructions (String, default: '') - Ghi chú đặc biệt
* paymentMethod (Enum, required) - Phương thức thanh toán: cash, bank\_transfer
* status (Enum, default: 'pending') - Trạng thái: pending, confirmed, preparing, ready, delivered, cancelled
* driverId (ObjectId, ref: Driver, optional) - **CẢI THIỆN**: ID tài xế (ref Driver thay vì User)
* estimatedDeliveryTime (Date, optional) - Thời gian giao hàng dự kiến
* actualDeliveryTime (Date, optional) - Thời gian giao hàng thực tế
* code (String, unique, optional) - Mã đơn hàng
* trackingHistory (Array, default: []) - **MỚI**: Lịch sử theo dõi:
* status (String) - Trạng thái
* timestamp (Date) - Thời gian
* note (String, optional) - Ghi chú
* updatedBy (String, optional) - Người cập nhật: system, restaurant, driver, customer

**7. OrderItem (Bảng chi tiết đơn hàng)**

* **Tên bảng**: orderitems
* **Mục đích**: Lưu trữ chi tiết từng món trong đơn hàng
* **Thuộc tính**:
* orderId (ObjectId, ref: Order, required) - ID đơn hàng
* itemId (ObjectId, optional) - ID món ăn
* nameSnapshot (String, optional) - Tên món ăn tại thời điểm đặt
* priceSnapshot (Number, optional) - Giá món ăn tại thời điểm đặt
* quantity (Number, optional) - Số lượng
* optionsSnapshot (Object, optional) - Tùy chọn món ăn

**8. Cart (Bảng giỏ hàng) - ✅ TỐI ƯU INDEX**

* **Tên bảng**: carts
* **Mục đích**: Quản lý giỏ hàng của khách hàng
* **Thuộc tính**:
* userId (ObjectId, ref: User, required) - ID khách hàng
* itemId (ObjectId, ref: Item, required) - ID món ăn
* restaurantId (ObjectId, ref: Restaurant, required) - ID nhà hàng
* quantity (Number, required, min: 1) - Số lượng
* specialInstructions (String, optional) - Ghi chú đặc biệt
* isActive (Boolean, default: true) - Trạng thái hoạt động

**9. Image (Bảng hình ảnh) - ✅ REFACTOR CLOUD STORAGE**

* **Tên bảng**: images
* **Mục đích**: Lưu trữ metadata hình ảnh (không lưu binary)
* **Thuộc tính**:
* filename (String, required) - Tên file
* originalName (String, required) - Tên gốc
* mimeType (String, required) - Loại MIME
* size (Number, required) - Kích thước (bytes)
* url (String, required) - **CẢI THIỆN**: URL cloud storage
* cloudProvider (String, optional) - **MỚI**: Provider: aws-s3, gcp, cloudinary
* cloudKey (String, optional) - **MỚI**: Key trong cloud storage
* uploadedBy (ObjectId, optional) - ID người upload
* type (String, default: 'image') - Loại ảnh: image, avatar, menu-item, restaurant
* isActive (Boolean, default: true) - Trạng thái hoạt động
* altText (String, optional) - **MỚI**: Text cho accessibility
* width (Number, optional) - **MỚI**: Chiều rộng
* height (Number, optional) - **MỚI**: Chiều cao

**�� QUẢN LÝ PHIÊN ĐĂNG NHẬP (1 bảng)**

**10. Session (Bảng phiên đăng nhập) - 🆕 MỚI**

* **Tên bảng**: sessions
* **Mục đích**: Quản lý phiên đăng nhập, refresh token, logout từ xa
* **Thuộc tính**:
* userId (ObjectId, ref: User, required) - ID người dùng
* refreshToken (String, unique) - Refresh token
* accessToken (String) - Access token
* status (Enum, default: 'active') - Trạng thái: active, expired, revoked, logged\_out
* expiresAt (Date) - Thời gian hết hạn
* lastUsedAt (Date) - Lần sử dụng cuối
* revokedAt (Date) - Thời gian thu hồi
* revokedBy (ObjectId) - Admin thu hồi
* revokedReason (String) - Lý do thu hồi
* userAgent (String) - Thông tin trình duyệt
* ipAddress (String) - Địa chỉ IP
* deviceId (String) - ID thiết bị
* deviceName (String) - Tên thiết bị
* platform (String) - Nền tảng: ios, android, web
* appVersion (String) - Phiên bản app
* isCurrentSession (Boolean) - Phiên hiện tại

**�� AUDIT & TRACKING (1 bảng)**

**11. AuditLog (Bảng nhật ký hoạt động) - 🆕 MỚI**

* **Tên bảng**: auditlogs
* **Mục đích**: Ghi lại tất cả hành vi quan trọng cho debug, compliance
* **Thuộc tính**:
* userId (ObjectId, ref: User) - Người thực hiện
* action (Enum, required) - Hành động: login, logout, order\_created, payment\_success, etc.
* level (Enum, default: 'info') - Mức độ: info, warning, error, critical
* description (String, required) - Mô tả hành động
* details (Object) - Chi tiết bổ sung
* oldValues (Object) - Giá trị cũ (cho updates)
* newValues (Object) - Giá trị mới (cho updates)
* ipAddress (String) - Địa chỉ IP
* userAgent (String) - Thông tin trình duyệt
* deviceId (String) - ID thiết bị
* platform (String) - Nền tảng
* orderId, restaurantId, driverId, paymentId - Entities liên quan
* targetUserId (ObjectId) - User bị ảnh hưởng
* sessionId (String) - ID phiên
* requestId (String) - ID request để tracing
* isVisible (Boolean, default: true) - Hiển thị (filter sensitive logs)

**💬 HỆ THỐNG CHAT (2 bảng)**

**12. Chat (Bảng cuộc trò chuyện) - �� MỚI**

* **Tên bảng**: chats
* **Mục đích**: Quản lý cuộc trò chuyện giữa các bên
* **Thuộc tính**:
* orderId (ObjectId, ref: Order) - ID đơn hàng
* type (Enum, required) - Loại: order\_support, customer\_support, restaurant\_support, driver\_support
* participants (Array[ObjectId]) - Người tham gia
* title (String) - Tiêu đề chat
* description (String) - Mô tả
* createdBy (ObjectId) - Người tạo
* lastMessageBy (ObjectId) - Người gửi tin nhắn cuối
* lastMessageAt (Date) - Thời gian tin nhắn cuối
* lastMessage (String) - Nội dung tin nhắn cuối
* isActive (Boolean, default: true) - Hoạt động
* isArchived (Boolean, default: false) - Đã lưu trữ

**13. Message (Bảng tin nhắn) - 🆕 MỚI**

* **Tên bảng**: messages
* **Mục đích**: Lưu trữ tin nhắn trong chat
* **Thuộc tính**:
* chatId (ObjectId, ref: Chat, required) - ID cuộc trò chuyện
* senderId (ObjectId, ref: User, required) - Người gửi
* receiverIds (Array[ObjectId]) - Người nhận (group chat)
* type (Enum, required) - Loại: text, image, file, system, order\_update, location, emoji
* content (String) - Nội dung text
* attachments (Array[String]) - URLs file đính kèm
* fileMetadata (Object) - Metadata file
* status (Enum, default: 'sent') - Trạng thái: sent, delivered, read, failed
* deliveredAt (Date) - Thời gian giao
* readAt (Date) - Thời gian đọc
* readBy (Array[ObjectId]) - Người đã đọc
* replyTo (ObjectId) - Tin nhắn được trả lời
* editedAt (Date) - Thời gian chỉnh sửa
* editedBy (ObjectId) - Người chỉnh sửa
* isEdited (Boolean, default: false) - Đã chỉnh sửa
* isDeleted (Boolean, default: false) - Đã xóa
* deletedAt (Date) - Thời gian xóa
* deletedBy (ObjectId) - Người xóa

**🎫 HỆ THỐNG HỖ TRỢ (1 bảng)**

**14. SupportTicket (Bảng ticket hỗ trợ) - 🆕 MỚI**

* **Tên bảng**: supporttickets
* **Mục đích**: Xử lý khiếu nại, phản hồi hỗ trợ
* **Thuộc tính**:
* ticketNumber (String, unique) - Số ticket: TK-2024-001234
* reporterId (ObjectId, ref: User) - Người báo cáo
* assigneeId (ObjectId, ref: User) - Người được phân công
* type (Enum, required) - Loại: order\_issue, payment\_issue, delivery\_issue, food\_quality, etc.
* status (Enum, default: 'open') - Trạng thái: open, in\_progress, pending\_customer, resolved, closed
* priority (Enum, default: 'medium') - Ưu tiên: low, medium, high, urgent, critical
* subject (String, required) - Tiêu đề
* description (String, required) - Mô tả
* attachments (Array[String]) - File đính kèm
* orderId, restaurantId, driverId, paymentId - Entities liên quan
* resolution (String) - Giải pháp
* resolvedAt (Date) - Thời gian giải quyết
* resolvedBy (ObjectId) - Người giải quyết
* closedAt (Date) - Thời gian đóng
* closedBy (ObjectId) - Người đóng
* closeReason (String) - Lý do đóng
* rating (Number, 1-5) - Đánh giá hỗ trợ
* feedback (String) - Phản hồi khách hàng
* firstResponseAt (Date) - Thời gian phản hồi đầu tiên
* resolutionDeadline (Date) - Hạn giải quyết
* escalationAt (Date) - Thời gian nâng cấp
* escalatedBy (ObjectId) - Người nâng cấp
* escalationReason (String) - Lý do nâng cấp
* internalNotes (String) - Ghi chú nội bộ
* tags (Array[String]) - Tags phân loại

**�� HỆ THỐNG THANH TOÁN & VÍ (3 bảng)**

**15. Payment (Bảng thanh toán) - �� MỚI**

* **Tên bảng**: payments
* **Mục đích**: Quản lý giao dịch thanh toán
* **Thuộc tính**:
* userId (ObjectId, ref: User, required) - ID người dùng
* orderId (ObjectId, ref: Order, required) - ID đơn hàng
* amount (Number, required) - Số tiền (VND)
* currency (String, required) - Đơn vị tiền tệ: VND, USD
* method (Enum, required) - Phương thức: cash, bank\_transfer, momo, vnpay, zalopay, credit\_card, debit\_card
* provider (Enum, required) - Nhà cung cấp: vnpay, momo, zalopay, stripe, paypal, manual
* status (Enum, default: 'pending') - Trạng thái: pending, processing, success, failed, cancelled, refunded, partially\_refunded
* transactionId (String) - ID giao dịch từ provider
* referenceCode (String) - Mã tham chiếu của hệ thống
* gatewayResponse (Object) - Phản hồi từ payment gateway
* failureReason (String) - Lý do thất bại
* paidAt (Date) - Thời gian thanh toán
* refundedAt (Date) - Thời gian hoàn tiền
* refundAmount (Number) - Số tiền hoàn
* refundReason (String) - Lý do hoàn tiền
* refundTransactionId (String) - ID giao dịch hoàn tiền
* description (String) - Mô tả
* metadata (Object) - Dữ liệu bổ sung
* webhookData (Object) - Dữ liệu webhook từ provider
* expiresAt (Date) - Thời gian hết hạn (pending payments)

**16. RestaurantPayout (Bảng thanh toán nhà hàng) - 🆕 MỚI**

* **Tên bảng**: restaurantpayouts
* **Mục đích**: Quản lý số tiền nhà hàng nhận được sau khi trừ phí
* **Thuộc tính**:
* restaurantId (ObjectId, ref: Restaurant, required) - ID nhà hàng
* processedBy (ObjectId, ref: User) - Admin xử lý
* amount (Number, required) - Số tiền (VND)
* currency (String, required) - Đơn vị tiền tệ
* method (Enum, required) - Phương thức: bank\_transfer, momo, vnpay, zalopay, cash
* status (Enum, default: 'pending') - Trạng thái: pending, processing, completed, failed, cancelled
* referenceCode (String) - Mã tham chiếu
* externalTransactionId (String) - ID giao dịch external
* bankAccount (Object) - Thông tin tài khoản ngân hàng
* eWalletAccount (Object) - Thông tin ví điện tử
* processedAt (Date) - Thời gian xử lý
* completedAt (Date) - Thời gian hoàn thành
* failureReason (String) - Lý do thất bại
* orderIds (Array[ObjectId]) - Orders liên quan
* orderCount (Number) - Số lượng orders
* grossAmount (Number) - Tổng tiền trước phí
* commissionRate (Number) - Tỷ lệ hoa hồng
* commissionAmount (Number) - Số tiền hoa hồng
* platformFee (Number) - Phí nền tảng
* processingFee (Number) - Phí xử lý
* otherFees (Number) - Phí khác
* netAmount (Number) - Số tiền cuối cùng
* settlementStartDate (Date) - Ngày bắt đầu kỳ thanh toán
* settlementEndDate (Date) - Ngày kết thúc kỳ thanh toán

**17. DriverPayout (Bảng thanh toán tài xế) - �� MỚI**

* **Tên bảng**: driverpayouts
* **Mục đích**: Quản lý thu nhập và phí giao hàng của tài xế
* **Thuộc tính**:
* driverId (ObjectId, ref: Driver, required) - ID tài xế
* processedBy (ObjectId, ref: User) - Admin xử lý
* amount (Number, required) - Số tiền (VND)
* currency (String, required) - Đơn vị tiền tệ
* method (Enum, required) - Phương thức thanh toán
* status (Enum, default: 'pending') - Trạng thái
* referenceCode (String) - Mã tham chiếu
* externalTransactionId (String) - ID giao dịch external
* bankAccount (Object) - Thông tin tài khoản ngân hàng
* eWalletAccount (Object) - Thông tin ví điện tử
* orderIds (Array[ObjectId]) - Orders liên quan
* orderCount (Number) - Số lượng orders
* deliveryFees (Number) - Phí giao hàng
* tips (Number) - Tiền tip
* bonuses (Number) - Thưởng
* incentives (Number) - Khuyến khích
* grossAmount (Number) - Tổng thu nhập
* platformFee (Number) - Phí nền tảng
* processingFee (Number) - Phí xử lý
* insuranceFee (Number) - Phí bảo hiểm
* otherDeductions (Number) - Khấu trừ khác
* netAmount (Number) - Số tiền cuối cùng
* totalDistance (Number) - Tổng quãng đường (km)
* totalDeliveryTime (Number) - Tổng thời gian giao (phút)
* averageRating (Number) - Đánh giá trung bình
* onTimeDeliveries (Number) - Số lần giao đúng giờ
* totalDeliveries (Number) - Tổng số lần giao

**🎯 HỆ THỐNG BUSINESS LOGIC (4 bảng)**

**18. Review (Bảng đánh giá) - 🆕 MỚI**

* **Tên bảng**: reviews
* **Mục đích**: Quản lý đánh giá và bình luận
* **Thuộc tính**:
* userId (ObjectId, ref: User, required) - ID người đánh giá
* restaurantId (ObjectId, ref: Restaurant, optional) - ID nhà hàng
* itemId (ObjectId, ref: Item, optional) - ID món ăn
* driverId (ObjectId, ref: Driver, optional) - ID tài xế
* orderId (ObjectId, ref: Order, optional) - ID đơn hàng
* type (Enum, required) - Loại: restaurant, item, driver
* rating (Number, required, min: 1, max: 5) - Điểm đánh giá
* comment (String) - Bình luận
* images (Array[String]) - URLs ảnh đánh giá
* isVerified (Boolean, default: false) - Xác thực mua hàng
* helpfulCount (Number, default: 0) - Số người thấy hữu ích
* isVisible (Boolean, default: false) - Hiển thị (moderation)
* response (String) - Phản hồi từ nhà hàng/tài xế
* responseAt (Date) - Thời gian phản hồi

**19. Notification (Bảng thông báo) - �� MỚI**

* **Tên bảng**: notifications
* **Mục đích**: Quản lý thông báo hệ thống
* **Thuộc tính**:
* userId (ObjectId, ref: User, required) - ID người nhận
* type (Enum, required) - Loại thông báo: order\_status\_update, order\_assigned, order\_ready, order\_delivered, order\_cancelled, promotion, system\_announcement, driver\_available, restaurant\_approved, payment\_success, payment\_failed
* title (String, required) - Tiêu đề
* body (String, required) - Nội dung
* data (Object) - Dữ liệu bổ sung cho deep linking
* isRead (Boolean, default: false) - Đã đọc
* readAt (Date) - Thời gian đọc
* priority (Enum, default: 'medium') - Mức độ: low, medium, high, urgent
* isVisible (Boolean, default: true) - Hiển thị
* scheduledAt (Date) - Thời gian lên lịch
* expiresAt (Date) - Thời gian hết hạn
* orderId, restaurantId, driverId - Entities liên quan

**20. Promotion (Bảng khuyến mãi) - �� MỚI**

* **Tên bảng**: promotions
* **Mục đích**: Quản lý voucher và khuyến mãi
* **Thuộc tính**:
* name (String, required) - Tên khuyến mãi
* description (String) - Mô tả
* code (String, required, unique) - Mã voucher
* type (Enum, required) - Loại: percentage, fixed\_amount, free\_delivery, buy\_x\_get\_y, minimum\_order
* status (Enum, default: 'draft') - Trạng thái: draft, active, paused, expired, cancelled
* value (Number, required) - Giá trị giảm giá
* maxDiscountAmount (Number) - Số tiền giảm tối đa (cho %)
* minOrderAmount (Number) - Đơn hàng tối thiểu
* maxOrderAmount (Number) - Đơn hàng tối đa
* startDate (Date, required) - Ngày bắt đầu
* endDate (Date, required) - Ngày kết thúc
* usageLimit (Number, default: 1) - Giới hạn sử dụng tổng
* usageLimitPerUser (Number, default: 1) - Giới hạn sử dụng mỗi user
* usedCount (Number, default: 0) - Số lần đã sử dụng
* targetUsers (Array[ObjectId], ref: User) - Users cụ thể
* applicableRestaurants (Array[ObjectId], ref: Restaurant) - Nhà hàng áp dụng
* applicableItems (Array[ObjectId], ref: Item) - Món ăn áp dụng
* target (Enum, default: 'all\_users') - Đối tượng: all\_users, new\_users, existing\_users, specific\_users, restaurant\_customers
* isActive (Boolean, default: true) - Hoạt động
* imageUrl (String) - URL ảnh khuyến mãi
* termsAndConditions (String) - Điều khoản
* createdBy (ObjectId) - Admin tạo
* metadata (Object) - Dữ liệu bổ sung

**21. DeliveryZone (Bảng khu vực giao hàng) - �� MỚI**

* **Tên bảng**: deliveryzones
* **Mục đích**: Quản lý khu vực giao hàng và phân công tài xế
* **Thuộc tính**:
* name (String, required) - Tên khu vực
* description (String) - Mô tả
* status (Enum, default: 'active') - Trạng thái: active, inactive, maintenance
* geometry (Object, required, 2dsphere index) - Hình dạng khu vực:
* type (String, default: 'Polygon') - Loại hình
* coordinates (Array[Array[Array[Number]]]) - Tọa độ polygon
* deliveryFee (Number, required) - Phí giao hàng cơ bản
* freeDeliveryThreshold (Number) - Ngưỡng miễn phí giao hàng
* estimatedDeliveryTime (Number, default: 30) - Thời gian giao hàng dự kiến (phút)
* maxDeliveryDistance (Number, default: 0) - Khoảng cách giao hàng tối đa (mét)
* assignedDrivers (Array[ObjectId], ref: Driver) - Tài xế được phân công
* driverCount (Number, default: 0) - Tổng số tài xế
* activeDriverCount (Number, default: 0) - Số tài xế hoạt động
* centerLatitude (Number) - Vĩ độ trung tâm
* centerLongitude (Number) - Kinh độ trung tâm
* city (String) - Thành phố
* district (String) - Quận/huyện
* ward (String) - Phường/xã
* isActive (Boolean, default: true) - Hoạt động
* metadata (Object) - Dữ liệu bổ sung

**❤️ HỆ THỐNG YÊU THÍCH (1 bảng)**

**22. Favorite (Bảng yêu thích) - 🆕 MỚI**

* **Tên bảng**: favorites
* **Mục đích**: Lưu nhà hàng/món ăn yêu thích của user
* **Thuộc tính**:
* userId (ObjectId, ref: User, required) - ID người dùng
* type (Enum, required) - Loại: restaurant, item, category
* restaurantId (ObjectId, ref: Restaurant, optional) - ID nhà hàng
* itemId (ObjectId, ref: Item, optional) - ID món ăn
* categoryId (ObjectId, ref: Category, optional) - ID danh mục
* notes (String) - Ghi chú cá nhân
* isActive (Boolean, default: true) - Hoạt động
* metadata (Object) - Dữ liệu bổ sung

**�� HỆ THỐNG PHÂN TÍCH & RECOMMENDATION (1 bảng)**

**23. SearchHistory (Bảng lịch sử tìm kiếm) - 🆕 MỚI**

* **Tên bảng**: searchhistories
* **Mục đích**: Log các query, click, conversion cho recommendation
* **Thuộc tính**:
* userId (ObjectId, ref: User, optional) - ID người dùng (optional cho anonymous)
* type (Enum, required) - Loại: restaurant, item, category, general
* source (Enum, required) - Nguồn: search\_bar, filter, category, recommendation, autocomplete, voice, qr\_code
* action (Enum, required) - Hành động: search, click, view, add\_to\_cart, order, favorite, share, rate, review
* query (String) - Từ khóa tìm kiếm
* filters (Object) - Bộ lọc áp dụng
* sortBy (String) - Tiêu chí sắp xếp
* sortOrder (String) - Thứ tự: asc, desc
* restaurantId, itemId, categoryId, orderId - Entities liên quan
* location (Object) - Vị trí tìm kiếm
* deviceInfo (Object) - Thông tin thiết bị
* sessionId (String) - ID phiên
* ipAddress (String) - Địa chỉ IP
* resultCount (Number, default: 0) - Số kết quả trả về
* position (Number, default: 0) - Vị trí item được click
* timeSpent (Number, default: 0) - Thời gian xem (giây)
* conversionValue (Number) - Giá trị chuyển đổi
* conversionAt (Date) - Thời gian chuyển đổi
* metadata (Object) - Dữ liệu bổ sung

**�� TỔNG KẾT CUỐI CÙNG**

**Phân loại bảng:**

* **Cấu trúc cơ bản**: 9 bảng (User, Driver, Restaurant, Category, Item, Order, OrderItem, Cart, Image)
* **Quản lý phiên**: 1 bảng (Session)
* **Audit & Tracking**: 1 bảng (AuditLog)
* **Hệ thống chat**: 2 bảng (Chat, Message)
* **Hệ thống hỗ trợ**: 1 bảng (SupportTicket)
* **Thanh toán & ví**: 3 bảng (Payment, RestaurantPayout, DriverPayout)
* **Business logic**: 4 bảng (Review, Notification, Promotion, DeliveryZone)
* **Yêu thích**: 1 bảng (Favorite)
* **Phân tích**: 1 bảng (SearchHistory)

**Tổng số bảng: 23 bảng**

**Tất cả đều có timestamps: createdAt, updatedAt**

**Hỗ trợ đầy đủ: Authentication, Authorization, Payment, Notification, Review, Promotion, Delivery tracking, Chat, Support, Analytics**

**Tài liệu tham khảo**

* Nguyen, T. (2023). *Challenges in Vietnam’s food delivery apps*. Journal of E-commerce Studies, 12(3), 45–53.
* Pham, L. (2023, October 15). Delivery drivers struggle with reduced income. *Tuoi Tre News*.
* Q&Me. (2024). *Vietnam Food Delivery Market Survey 2024*. Retrieved from https://qandme.net
* Statista. (2024). *Online food delivery market in Vietnam*. Retrieved from https://www.statista.com
* Tien Phong. (2024, June 2). Nỗi khổ shipper thời công nghệ. *Tien Phong Online*.
* VnExpress. (2024, April 11). Nhiều nhà hàng kêu trời vì chiết khấu cao. *VnExpress*.
* Phan, V. (2024). *Personal field notes from over two years working as a delivery driver*. Unpublished manuscript.