BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

LẬP TRÌNH THIẾT BỊ DI ĐỘNG

BÁO CÁO CUỐI KỲ

ỨNG DỤNG ĐẶT TRÀ SỮA CHANGTEA

Giảng viên hướng dẫn: ThS. Huỳnh Tuấn Anh

Sinh viên thực hiện: **Phạm Tuấn Kiệt**

Cao Linh Hà

Nguyễn Hiểu Quyên

Nguyễn Hồ Thanh Bình

Khánh Hòa – 2025

DANH SÁCH ĐÓNG GÓP VÀ MỨC ĐỘ THAM KHẢO

Bảng 1.1. Danh sách thành viên và các đóng góp

STT	Thành viên	Mức độ	Chi tiết
1	Phạm Tuấn Kiệt	30%	- Tìm hiểu về Supabase - Mô tả các yêu cầu/chức năng - Thiết kế CSDL - Thiết kế Model/Controller Món, Giỏ Hàng. - Thiết kế giao diện và chức năng của trang chủ, trang chi tiết món, danh sách món, giỏ hàng, tìm kiếm. - Tổng hợp và viết báo cáo.
2	Cao Linh Hà	25%	- Tìm hiểu về Android Studio. - Đánh giá/khảo sát thực trạng. - Thiết kế CSDL. - Thiết kế Model/Controller hóa đơn, chi tiết hóa đơn. - Thiết kế giao diện và các chức năng của trang hóa đơn, thanh toán, lịch sử mua và chi tiết đơn. - Chỉnh sửa bài báo cáo.
3	Nguyễn Hồ Thanh Bình	25%	- Tìm hiểu về Flutter và ngôn ngữ Dart. - Tìm hiểu về công cụ quản lý trạng thái - Thiết kế CSDL. - Thiết kế Model/Controller khách hàng. - Thiết kế giao diện và chức năng của trang đăng nhập/ đăng xuất.
4	Nguyễn Hiểu Quyên	20%	- Tổng quan về quán trà sữa ChangTea.

			- Tìm hiểu về các PlugIn sử dụng - Thiết kế CSDL. - Thiết kế giao diện và chức năng của trang hiển thị, chỉnh sửa trang cá nhân.

Bảng 1.2. Mức độ tham khảo từ nguồn bên ngoài

STT	Nguồn tham khảo	Tỉ lệ	Cách thức tham khảo	Phần tham khảo
1	Các bài tập trên lớp	50%	- Tham khảo cách tổ chức model/controller.	- Các hàm cơ bản
2	Supabase Docs	20%	- Tham khảo cách tổ chức dữ liệu. - Cách sử dụng Supabase	- Các hàm thao tác với Supabase - Chức năng đăng nhập với mã xác nhận OTP
3	pub.dev	20%	Các thư viện hỗ trợ	
4	Chat GPT	10%	- Tham khảo cách chỉnh sửa. - Tham khảo ý tưởng	

MỤC LỤC

DANH MỤC HÌNH VẼ, ĐỒ THỊ

DANH MỤC BẢNG BIỂU

Chương 1. TỔNG QUAN QUÁN TRÀ SỮA CHANGTEA

1.1 GIỚI THIỆU CHUNG VỀ QUÁN

ChangTea là quán trà sữa do chị Dương Quỳnh Ngọc – 26 tuổi làm chủ, quán hoạt động từ 8h30 đến 22h mỗi ngày. Quán trà sữa ChangTea chính thức được thành lập vào ngày 20/01/2022, với mong muốn ban đầu của chị là mang đến cho khách hàng những trải nghiệm tuyệt vời về các loại trà sữa độc đáo và chất lượng. Sau quá trình chuẩn bị kỹ lưỡng, quán đã đi vào hoạt động vào ngày 15/03/2022, đánh dấu cột mốc quan trọng trong chặng đường kinh doanh của mình. Tính đến nay, quán đã kinh doanh được hơn 2 năm.

Hình 1.1. Logo quán trà sữa ChangTea

Ngay từ khi khai trương, ChangTea đã nhận được sự ủng hộ nhiệt tình từ khách hàng nhờ vào không gian thoải mái và sản phẩm chất lượng. Đến nay, ChangTea đã mở rộng quy mô với hai chi nhánh, bao gồm chi nhánh đầu tiên tại Bình Dương (119 KP Khánh Hòa, P. Tân Phước Khánh, Tân Uyên) và chi nhánh thứ hai ở Nha Trang (tại địa chỉ 15 Lam Sơn, Phước Hòa). Sắp tới, quán sẽ tiếp tục phát triển với việc khai trương chi nhánh thứ hai tại Nha Trang, đánh dấu bước phát triển vững chắc trong chiến lược mở rộng thị trường.

Ban đầu khi mới mở quán, lượng khách đến quán trung bình mỗi ngày chỉ dao động từ 30 đến 40 khách. Ngoài ra, nhân viên chỉ từ 2 – 3 bạn làm các công việc thay phiên nhau, quy mô quán còn khá nhỏ. Thế nhưng đến nay, trung bình quán tiếp đón hơn 100 khách mỗi ngày. Vào giờ cao điểm, lượng khách có thể vượt quá 200 người, đặc biệt là vào các ngày đầu tuần chưa kể ngày lễ. Hiện tại, với lượng khách ngày càng tăng, đội ngũ nhân viên đã tăng lên đến 9 – 10 người. Với nhu cầu phát triển không ngừng, quán dự kiến sẽ mở rộng quy mô nhân sự để phục vụ khách hàng tốt hơn, đặc biệt khi chuẩn bị ra mắt thêm chi nhánh mới.

ChangTea không ngừng nỗ lực để cải thiện và phát triển, nhằm đáp ứng nhu cầu ngày càng tăng của khách hàng và khẳng định vị thế trên thị trường trà sữa cạnh tranh.

1.2 MỤC TIÊU ĐỀ RA

Tạo ra một ứng dụng di động có thể giúp khách hàng dễ dàng tiếp cận với các món của quán. Ứng dụng hiển thị đầy đủ thông tin của các món nước và món ăn kèm với những thông tin cơ bản như giá, kích thức, món ăn kèm, mô tả và đánh giá. Giúp cho khách hàng dù ở xa hay gần thì vẫn có thể thưởng thức những món của quán. Bên cạnh đó, ứng dụng còn giúp lưu thông tin của khách hàng như tên, số điện thoại và địa chỉ thuận tiện cho việc giao hàng.

Chương 2. CƠ SỞ LÝ THUYẾT

2.1 TÌM HIỂU VỀ ANDROID STUDIO

2.1.1 Android Studio là gì?

Android Studio là môi trường phát triển tích hợp (IDE – Integrated Development Environment) chính thức của Google, được thiết kế dành riêng cho phát triển ứng dụng Android. Công cụ mạnh mẽ này cung cấp nhiều tính năng giúp các nhà phát triển tạo ra các ứng dụng chất lượng cao.

Android Studio được phát triển dựa trên nền tảng IntelliJ IDEA và được tích hợp nhiều công cụ để phát triển, thử nghiệm và gỡ lỗi ứng dụng Android. Ngoài ra IDE cũng được tích hợp trình giả lập Android, cho phép các nhà phát triển thử nghiệm ứng dụng của họ trên các thiết bị và kích thước màn hình khác nhau.

2.1.2 Các tính năng của Android Studio [1]

2.1.2.1 Môi trường phát triển tích hợp (IDE) mạnh mẽ

Là IDE chính thức để phát triển các ứng dụng Android, Android Studio được đóng gói đầy đủ các tính năng, bao gồm bộ công cụ như Code Editor, Debugging, Build System và Emulator. Bên cạnh đó, IDE cũng được cập nhật thường xuyên, đảm bảo các nhà phát triển có quyền truy cập vào các công cụ và công nghệ mới nhất.

2.1.2.2 Trình giả lập Android

Trình giả lập Android được tích hợp trên Android Studio cho phép các nhà phát triển thử nghiệm ứng dụng của họ trên các thiết bị ảo khác nhau với nhiều kích thước màn hình, độ phân giải và phiên bản Android khác nhau. Ngoài ra, các nhà phát triển có thể kết nối các thiết bị vật lý để thử nghiệm, cho phép tạo ra môi trường thử nghiệm toàn diện.

2.1.2.3 Công cụ thiết kế giao diện người dùng (UI Designer)

Trình chỉnh sửa bố cục XML của Android Studio giúp đơn giản hóa quy trình thiết kế giao diện người dùng. Các nhà phát triển có thể kéo và thả các thành phần UI, xem trước bố cục theo thời gian thực và quản lý hiệu quả các khía cạnh trực quan của ứng dụng.

2.1.2.4 Hỗ trợ đa nền tảng

Android Studio hỗ trợ phát triển ứng dụng cho mọi kích thước màn hình, các thiết bị đeo và thiết bị thông minh. Chẳng hạn như điện thoại, máy tính bảng, TV, đồng hồ thông minh, xe hơi (Android Auto) và các thiết bị gia đình thông minh. Ngoài ra công cụ cũng có thể mô phỏng nhiều loại tính năng khác nhau mà phần cứng có như trình theo dõi vị trí GPS, multi-touch.

2.1.2.5 Quản lý và tối ưu hóa mã

Android Studio - IDE chính thức từ Google tích hợp trình soạn thảo code tiên tiến, hỗ trợ đa dạng ngôn ngữ với trọng tâm là Java và Kotlin. Nổi bật với tính năng gợi ý code thông minh, phát hiện lỗi tức thì và công cụ phân tích mã nguồn tự động, giúp nâng cao chất lượng, hiệu năng và bảo mật cho ứng dụng ngay trong quá trình phát triển.

2.1.2.6. Công cụ hỗ trợ kiểm tra và debug

Nền tảng tích hợp bộ công cụ debug mạnh mẽ, giúp lập trình viên dễ dàng phát hiện và sửa lỗi trong quá trình phát triển ứng dụng Android. Hệ thống gỡ lỗi thông minh cho phép theo dõi luồng thực thi, kiểm tra biến và tối ưu hiệu suất chương trình một cách nhanh chóng.

2.1.3 Ưu, nhược điểm

- Ưu điểm:

+ Hỗ trợ nhiều ngôn ngữ lập trình: Java, C++,…
+ Cung cấp nhiều thiết bị ảo (AVD) để kiểm tra ứng dụng trên nhiều kích thước màn hình và cấu hình khác nhau.
+ Tích hợp nhiều công cụ như Debugger,.. giúp phát hiện lỗi và tối ưu dễ dàng.
+ Dễ dàng tích hợp Firebase Analytics, Authentication,…
+ Thường xuyên cập nhật phiên bản mới với nhiều tính năng cải tiến và sữa lỗi.

- Nhược điểm:

+ Công cụ chiếm lượng lớn dữ liệu trong không gian bộ nhớ máy tính nên yêu cầu cấu hình máy cao.
+ Việc kiểm tra hoạt động thông qua giả lập gây đơ, lag, giật máy và tiêu tốn pin.

+ Tốc độ khở dộng và build chậm, đặc biệt khi chạy nhiều thiết bị ảo haowjc dự án có nhiều dependencies.

2.2 TÌM HIỂU VỀ FLUTTER

2.2.1 Flutter là gì?

Hình 2.1. Framework Flutter

Flutter là Framework nguồn mở ra mắt vào năm 2018 do Google phát triển và hỗ trợ. Các nhà phát triển frontend và fullstack sử dụng Flutter để xây dựng giao diện người dùng (User Interface) của ứng dụng cho nhiều nền tảng chỉ với một nền mã duy nhất. Tại thời điểm ra mắt, Flutter chủ yếu hỗ trợ phát triển ứng dụng di động. Hiện nay, Flutter hỗ trợ phát triển ứng dụng trên đa nền tảng như IOS, Android, Web, Windows, MacOS và Linux [2].

2.2.2 Các thành phần chính của Flutter

2.2.2.1 Flutter SDK

Flutter SDK (Flutter Software Development Kit) [3] là một bộ công cụ phát triển phần mềm cung cấp các thư viện và công cụ cần thiết để xây dựng các ứng dụng Flutter. Flutter SDK bao gồm:

Bảng 2.1. Các thành phần của Flutter SDK

STT	Thành phần	Chức năng
1	Dart SDK	Dart là ngôn ngữ lập trình được sử dụng để viết các ứng dụng Flutter. Dart SDK cung cấp trình biên dịch và các công cụ cần thiết để làm việc với Dart.

2	Flutter Engine	Flutter Engine là lõi của Flutter, được viết bằng C++, cung cấp các công cụ cần thiết để render giao diện người dùng và thực hiện các tác vụ như xử lý sự kiện, vẽ đồ họa và quản lý các thành phần UI.
3	Framework	Framework của Flutter được viết bằng Dart, bao gồm các thư viện và widget để xây dựng giao diện người dùng, quản lý trạng thái, điều hướng và nhiều chức năng khác.

2.2.2.2 Widgets

Widgets [3] là thành phần cơ bản của giao diện người dùng trong Flutter. Tất cả mọi thứ trong Flutter đều được coi như một Widgets. Flutter cung cấp cho người dùng 2 loại Widget chính.

Bảng 2.2. Hai loại Widget chính của Flutter

STT	Thành phần	Chức năng
1	Stateless Widgets	Widgets không thay đổi trạng thái trong suốt vòng đời của chúng. Chúng chỉ phụ thuộc vào dữ liệu đầu vào và hiển thị giao diện người dùng dựa trên đó.
2	Stateful Widgets	Widgets có thể thay đổi trạng thái trong suốt vòng đời của chúng. Chúng có thể phản ứng với các sự kiện và cập nhật giao diện người dùng dựa trên trạng thái hiện tại.

2.2.2.3 Flutter DevTools

Flutter DevTools [3] là bộ công cụ phát triển tích hợp được cung cấp để giúp các nhà phát triển gỡ lỗi, phân tích và tối ưu hóa các ứng dụng Flutter. Bộ công cụ này bao gồm:

Bảng 2.3. Các công cụ của DevTools

STT	Thành phần	Chức năng
1	Inspector	Công cụ để kiểm tra và chỉnh sửa cấu trúc widget của ứng dụng.
2	Logging	Công cụ để xem và phân tích log từ ứng dụng.

3	Performance	Công cụ để phân tích hiệu suất của ứng dụng, bao gồm theo dõi khung hình và phân tích CPU.

2.2.2.4 Packages and Plugins

Flutter hỗ trợ việc sử dụng các package và plugin để mở rộng chức năng của ứng dụng. Các package là các thư viện Dart có thể được chia sẻ và sử dụng lại, trong khi các plugin cung cấp giao diện để tương tác với mã gốc (Native Code) trên các nền tảng iOS và Android.

2.2.3 Một số ứng dụng được phát triển từ Flutter

Được trang bị nhiều tiện ích, hiệu suất đáng chú ý và các thư viện mã nguồn mở toàn diện, Flutter nổi lên như một lựa chọn tối ưu để xây dựng các ứng dụng ấn tượng về mặt hình ảnh và lấy người dùng làm trung tâm [4].

Bảng 2.4. Các ứng dụng hàng đầu sử dụng Flutter

STT	Ứng dụng	Chức năng
1	BMW App	Ứng dụng di động chính thức do hãng BMW phát triển, giúp chủ sở hữu xe BMW quản lý xe của mình dễ dàng hơn và tận hưởng các tiện ích kết nối hiện đại.
2	Toyota App	Ứng dụng di động chính thức do Toyota phát triển, nhằm hỗ trợ người dùng sở hữu và sử dụng xe Toyota một cách tiện lợi hơn. Ứng dụng này cung cấp nhiều tính năng liên quan đến xe, dịch vụ hậu mãi, và kết nối giữa người dùng với hãng Toyota.
3	Alibaba	Ứng dụng di động chính thức của Alibaba.com, nền tảng thương mại điện tử B2B (Business to Business – doanh nghiệp với doanh nghiệp) lớn nhất thế giới do tập đoàn Alibaba (Trung Quốc) phát triển.
4	Google Pay	Ứng dụng và nền tảng thanh toán điện tử do Google phát triển, cho phép người dùng thanh toán không chạm (contactless), chuyển tiền, mua sắm trực tuyến, và quản lý

		tài chính một cách tiện lợi bằng điện thoại di động hoặc máy tính.

2.3 TÌM HIỂU VỀ NGÔN NGỮ DART

2.3.1 Ngôn ngữ lập trình Dart

Dart là ngôn ngữ lập trình hướng đối tượng, mục đích chung do Google phát triển, lần đầu tiên được giới thiệu vào năm 2011. Dart có cú pháp tương tự như ngôn ngữ lập trình C và được biết đến với tốc độ biên dịch nhanh, cú pháp dễ hiểu và khả năng phát triển giao diện người dùng linh hoạt. Tính linh hoạt của Dart được chứng minh thông qua khả năng xây dựng các ứng dụng web, di động (Android và iOS), máy chủ và máy tính để bàn, nhờ tích hợp với Flutter, một khuôn khổ phổ biến của Google. Điều này cung cấp cho các nhà phát triển nhiều tùy chọn để tạo các ứng dụng hiện đại và hiệu quả [5].

2.3.2 Một số đặc điểm nổi bật [5]

2.3.2.1 Cú pháp đơn giản và dễ đọc

Dart sở hữu cú pháp tương đồng với các ngôn ngữ lập trình theo phong cách C như Java và JavaScript, giúp các lập trình viên dễ dàng tiếp cận và nhanh chóng làm quen. Điều này góp phần rút ngắn thời gian học tập và cho phép bắt đầu phát triển ứng dụng trong thời gian ngắn.

2.3.2.2 Hỗ trợ lập trình hướng đối tượng

Dart là một ngôn ngữ hướng đối tượng đầy đủ tính năng, hỗ trợ các khái niệm như kế thừa theo mô hình lớp (class-based inheritance) và đa hình (polymorphism). Điều này cho phép tổ chức mã nguồn một cách rõ ràng, dễ bảo trì và tái sử dụng. Mô hình OOP còn tạo điều kiện thuận lợi cho việc phát triển các ứng dụng có kiến trúc phức tạp.

2.3.2.3 Tính năng Null Safety

Một trong những tính năng hiện đại và quan trọng của Dart là null safety – giúp ngăn chặn các lỗi thực thi (runtime errors) liên quan đến giá trị null bằng cách đảm bảo rằng các biến không thể mang giá trị null nếu không được khai báo rõ ràng. Null safety góp phần nâng cao độ an toàn của mã nguồn và giúp lập trình viên tránh được các lỗi phổ biến, đặc biệt hữu ích trong các dự án lớn có cấu trúc phức tạp.

2.3.2.4 Khả năng biên dịch nhanh

Dart hỗ trợ cả hai cơ chế biên dịch là Ahead-Of-Time (Một phương pháp biên dịch mã nguồn thành mã máy gốc trước khi ứng dụng được khởi chạy) và Just-In-Time (Một kỹ thuật biên dịch trung gian thành mã máy ngay trong quá trình ứng dụng đang chạy), tối ưu hóa quy trình phát triển ứng dụng. Ngoài việc biên dịch sang mã máy gốc để chạy hiệu quả trên thiết bị di động và máy tính để bàn, Dart còn có thể biên dịch sang mã JavaScript để chạy trên trình duyệt web. Khả năng biên dịch nhanh này giúp rút ngắn thời gian chờ đợi, tăng năng suất làm việc và cải thiện trải nghiệm phát triển phần mềm.

2.3.2.5 Phát triển đa nền tảng (Cross-platform)

Khi kết hợp với Framework Flutter, ngôn ngữ Dart cho phép xây dựng ứng dụng đa nền tảng từ một cơ sở mã duy nhất. Lập trình viên có thể triển khai ứng dụng cho iOS, Android, web và máy tính để bàn mà không cần viết mã riêng biệt cho từng nền tảng. Điều này giúp tiết kiệm tài nguyên và thời gian phát triển, đồng thời đảm bảo tính nhất quán và chất lượng giữa các nền tảng.

2.3.2.6 Tính năng Hot Reload

Một trong những tính năng nổi bật nhất của Dart (khi sử dụng với Flutter) là hot reload – cho phép lập trình viên quan sát ngay lập tức các thay đổi trong mã nguồn mà không cần khởi động lại ứng dụng. Điều này rút ngắn chu trình phát triển – kiểm thử, giúp quá trình xây dựng và gỡ lỗi trở nên nhanh chóng và hiệu quả hơn.

2.3.2.7 Khả năng tích hợp tốt với công cụ phát triển hiện đại

Dart tích hợp hiệu quả với nhiều môi trường phát triển tích hợp (IDE) như Visual Studio Code, Android Studio và IntelliJ IDEA. Các công cụ này cung cấp các tính năng như tự động hoàn thiện mã (auto-completion), kiểm tra lỗi thời gian thực (real-time error checking), và quản lý dự án, giúp nâng cao năng suất và hiệu quả công việc.

2.4 TÌM HIỂU VỀ NỀN TẢNG SUPABASE

2.4.1 Tổng quan về Supabase

Hình 2.2. Nền tảng Supabase

Supabase được thành lập vào tháng 1 năm 2020 bởi Paul Copplestone và Ant Wilson, xuất phát từ nhu cầu tìm giải pháp thay thế Firebase do những hạn chế về tốc độ và khả năng truy vấn. Ban đầu gặp khó khăn trong việc thu hút người dùng, bước ngoặt đến khi họ định vị Supabase là "giải pháp mã nguồn mở thay thế Firebase", giúp nền tảng nhanh chóng thu hút cộng đồng lập trình viên và tăng trưởng mạnh mẽ. Supabase tham gia chương trình Y Combinator mùa hè 2020 và đến tháng 7 cùng năm đã lưu trữ hơn 3.000 cơ sở dữ liệu. Đến tháng 4 năm 2024, Supabase mua lại OrioleDB nhằm nâng cao hiệu năng cơ sở dữ liệu, bổ sung công nghệ từ chuyên gia PostgreSQL Alexander Korotkov.

Supabase là một giải pháp thay thế nguồn mở cho Firebase, nó cung cấp một bộ công cụ và dịch vụ cho các nhà phát triển để xây dựng và mở rộng ứng dụng nhanh chóng mà không cần quản lý cơ sở hạ tầng máy chủ. Điều này bao gồm nhiều tính năng, chẳng hạn như cơ sở dữ liệu Postgres (Postgres Database), xác thực người dùng (User Authentication), API tức thời (Instant APIs), chức năng không có máy chủ (Serverless Functions), lưu trữ tệp (File Storage) và đồng bộ hóa dữ liệu theo thời gian thực (Real-Time Data Synchronization).

Một trong những điểm khác biệt chính của Supabase là bản chất nguồn mở và nền tảng của nó trong Postgres, một cơ sở dữ liệu quan hệ mạnh mẽ và linh hoạt. Điều này mang lại một số lợi thế so với các giải pháp BaaS (Backend-as-a-Service là mô hình dịch vụ đám mây trong đó các nhà phát triển thuê ngoài tất cả các khía cạnh hậu trường của ứng dụng web hoặc di động để họ chỉ phải viết và duy trì giao diện người dùng) độc quyền như Firebase, bao gồm tính minh bạch, tính linh hoạt và hiệu quả về chi phí được tăng cường [6].

2.4.2 Các tính năng chính của Supabase

2.4.2.1 Cơ sở dữ liệu PostgreSQL

PostgreSQL là một hệ thống cơ sở dữ liệu quan hệ đối tượng mã nguồn mở mạnh mẽ sử dụng và mở rộng ngôn ngữ SQL kết hợp với nhiều tính năng lưu trữ và mở rộng an toàn các khối lượng công việc dữ liệu phức tạp nhất [7].

Supabase cung cấp cơ sở dữ liệu PostgreSQL được quản lý hoàn toàn, cho phép nhà phát triển truy cập và thao tác dữ liệu thông qua các API tức thời (Instant APIs) được tự động sinh ra. Hệ thống hỗ trợ các truy vấn nâng cao, bao gồm tìm kiếm toàn văn (full-text search) và xử lý các kiểu dữ liệu phức tạp như JSON, giúp quản lý dữ liệu linh hoạt và hiệu quả hơn. Đồng thời, Supabase áp dụng cơ chế bảo mật trên từng hàng dữ liệu (row-level security), cho phép kiểm soát quyền truy cập một cách chi tiết và chặt chẽ, đảm bảo an toàn dữ liệu tối ưu [8].

2.4.2.2 Xác thực – Authentication [6]

Supabase cung cấp một hệ thống quản lý người dùng toàn diện với đa dạng phương thức xác thực, bao gồm xác thực bằng email và mật khẩu, liên kết phép thuật (magic links), đăng nhập xã hội (social logins) và đăng nhập một lần (SSO). Hệ thống này đơn giản hóa quy trình đăng ký và quản lý người dùng một cách hiệu quả.

Cho phép bảo mật theo hàng (Row-Level Security), kiểm soát quyền truy cập dữ liệu ở mức từng bản ghi dựa trên vai trò và thuộc tính của người dùng, từ đó đảm bảo tính bảo mật và riêng tư của dữ liệu. Cơ chế này cho phép thực thi các chính sách phân quyền chi tiết và tách biệt dữ liệu giữa các nhóm người dùng.

Triển khai các chính sách kiểm soát truy cập chi tiết bằng cách tận dụng khả năng của Row-Level Security trong PostgreSQL. Điều này giúp các nhà phát triển xác định rõ ràng ai được phép truy cập dữ liệu nào và với mức độ quyền hạn ra sao, góp phần tăng cường bảo mật và quản lý dữ liệu hiệu quả.

2.4.2.3 Đồng bộ hóa dữ liệu theo thời gian thực

Supabase cho phép đồng bộ dữ liệu theo thời gian thực bằng cách tận dụng các tính năng sao chép (replication) của PostgreSQL. Nhờ đó, các ứng dụng có thể được xây dựng với khả năng cộng tác trực tiếp, phản hồi ngay lập tức các thay đổi xảy ra trong cơ sở dữ liệu. Điều này tạo điều kiện thuận lợi cho việc phát triển các ứng dụng tương tác

cao, nơi người dùng có thể cùng thao tác và cập nhật dữ liệu đồng bộ mà không bị trễ [8].

2.4.2.4 Lưu từ - Storage

Supabase cung cấp hệ thống lưu trữ tệp tích hợp, cho phép lưu trữ, tổ chức và phân phối các tệp dung lượng lớn như hình ảnh và video. Đây là một giải pháp tập trung và có khả năng mở rộng cao, phù hợp cho việc quản lý tài nguyên số của ứng dụng. Thông qua việc tích hợp với mạng phân phối nội dung (CDN Integration), Supabase hỗ trợ phân phối tệp nhanh chóng và hiệu quả bằng cách lưu vào bộ đệm tại các điểm gần người dùng, từ đó giảm độ trễ truy cập và nâng cao hiệu suất tổng thể của ứng dụng. Hệ thống hỗ trợ xử lý hình ảnh trực tiếp, bao gồm thay đổi kích thước, cắt xén và tối ưu hóa hình ảnh theo thời gian thực. Điều này không chỉ đơn giản hóa quá trình quản lý hình ảnh mà còn giúp tiết kiệm không gian lưu trữ và băng thông sử dụng [6].

2.4.2.5 Chức năng máy chủ biên - Edge Functions

Supabase hỗ trợ các hàm không máy chủ (serverless functions), cho phép thực thi mã tùy chỉnh mà không cần quản lý hạ tầng máy chủ. Mô hình này phù hợp với kiến trúc không máy chủ và các hệ thống vi dịch vụ (microservices), giúp giảm thiểu chi phí vận hành và đơn giản hóa quy trình triển khai.

Các hàm có thể được triển khai tại các điểm nút gần với người dùng, giúp giảm độ trễ và nâng cao hiệu suất thực thi. Việc phân phối toàn cầu (Global Distribution) này góp phần cải thiện trải nghiệm người dùng thông qua tốc độ phản hồi nhanh hơn.

Các hàm được viết bằng TypeScript, tận dụng tính năng kiểm tra kiểu tĩnh (type safety) và các công cụ thân thiện với lập trình viên. Điều này giúp nâng cao chất lượng mã nguồn và giảm thiểu lỗi trong quá trình phát triển [6].

2.5 CÁC PLUGIN ĐƯỢC SỬ DỤNG

2.5.1 Dart

Dart Plugin là một tiện ích mở rộng dành cho các IDE như VS Code, Android Studio, và IntelliJ IDEA, giúp lập trình viên viết mã với ngôn ngữ Dart một cách hiệu quả hơn, nhanh hơn, và chính xác hơn. Dart Plugin kết nối với Dart SDK để phân tích cú pháp, gợi ý mã, và chạy ứng dụng. Sử dụng Language Server Protocol (LSP) để hỗ

trợ các tính năng thông minh như gợi ý, phân tích, refactor... Kết hợp với Flutter Plugin để hỗ trợ toàn bộ hệ sinh thái Flutter.

Các tính năng chính của Dart Plugin bao gồm: hỗ trợ mã hóa thông minh với khả năng gợi ý mã, định dạng tự động, điều hướng mã nguồn, đề xuất cải tiến (intentions), tái cấu trúc mã (refactorings), và nhiều tiện ích khác. Công cụ này tích hợp chặt chẽ với hệ thống quản lý gói Pub và Dart Analysis Server, cho phép phân tích mã theo thời gian thực, phát hiện lỗi tức thì và đưa ra gợi ý sửa lỗi. Ngoài ra, nó còn cung cấp trình gỡ lỗi (debugger) tích hợp để hỗ trợ gỡ lỗi các ứng dụng Dart dòng lệnh và web. Hệ thống cũng cho phép chạy và kiểm thử đơn vị (unit test) cho mã Dart. Người dùng có thể dễ dàng khởi tạo dự án mới thông qua tùy chọn có sẵn ngay tại màn hình chào mừng [9].

2.5.2 Flutter

Flutter Plugin cung cấp cho các nhà phát triển một cách dễ dàng và hiệu quả để xây dựng và triển khai các ứng dụng di động đa nền tảng, hiệu suất cao cho cả Android và iOS [10].

Các tính năng nổi bật Flutter Plugin bao gồm: khả năng tạo dự án Flutter mới thông qua giao diện đơn giản với các mẫu (template) có sẵn, giúp khởi tạo nhanh chóng. Công cụ hỗ trợ tự động gợi ý mã (autocomplete), đặc biệt hiệu quả với các widget phổ biến như Container, Text, Scaffold, giúp tăng tốc độ lập trình. Tính năng Hot Reload và Hot Restart cho phép cập nhật giao diện ngay lập tức khi thay đổi mã nguồn mà không cần khởi động lại ứng dụng, nâng cao hiệu suất phát triển. Người dùng có thể trực tiếp chạy và gỡ lỗi ứng dụng Flutter trên thiết bị thật hoặc giả lập, đồng thời tận dụng Dart DevTools để theo dõi hiệu năng, bộ nhớ, log và mạng. Ngoài ra, công cụ cũng hỗ trợ refactor nhanh các widget, chẳng hạn như tách một widget thành một lớp riêng biệt, giúp tổ chức mã rõ ràng hơn. Cuối cùng, hệ thống log tích hợp giúp dễ dàng theo dõi và xử lý lỗi giao diện từ emulator hoặc thiết bị vật lý.

2.5.3 Flutter Intl

Flutter Intl Plugin là một plugin hỗ trợ quốc tế hóa (internationalization - i18n) trong các dự án Flutter, giúp quản lý đa ngôn ngữ một cách tự động và thuận tiện. Đây là công cụ phép tạo và quản lý các tệp Application Resource Bundle chứa các bản dịch văn bản theo từng ngôn ngữ [11].

Flutter Intl Plugin cung cấp nhiều tính năng mạnh mẽ hỗ trợ quá trình quốc tế hóa trong dự án Flutter. Plugin cho phép tạo các tệp .arb riêng biệt cho từng ngôn ngữ, trong đó mỗi tệp chứa các cặp key-value dùng để dịch văn bản. Theo chuẩn Flutter i18n, plugin tự động sinh các tệp như intl_en.arb, intl_vi.arb,... để quản lý bản dịch. Đồng thời, nó tạo ra lớp S giúp truy cập chuỗi ngôn ngữ một cách dễ dàng thông qua cú pháp như S.of(context).hello. Một điểm nổi bật khác là khả năng hỗ trợ hot reload, cho phép cập nhật bản dịch mà không cần khởi động lại ứng dụng. Plugin cũng hỗ trợ đồng bộ key-value giữa các ngôn ngữ, đồng thời cảnh báo nếu thiếu bản dịch ở ngôn ngữ nào đó. Ngoài ra, nhờ tích hợp sâu với Android Studio, người dùng có thể thao tác nhanh với giao diện trực quan để thêm, chỉnh sửa các key/value một cách thuận tiện.

2.5.4 Dart Data Class

Dart Data Class Plugin là data model được sử dụng để biểu diễn các đối tượng chứa thông tin trong ứng dụng Flutter. Lớp này thường dùng để lưu trữ, truyền dữ liệu giữa các thành phần hoặc ánh xạ dữ liệu từ định dạng JSON. Dart Data Class là một class đơn giản, bao gồm các thành phần cơ bản như: các thuộc tính (fields) để lưu trữ giá trị, hàm khởi tạo (constructor) để tạo đối tượng, cùng các phương thức tiện ích như fromJson và toJson để chuyển đổi dữ liệu giữa đối tượng và JSON, copyWith để tạo bản sao có chỉnh sửa, và các phương thức ghi đè == cùng hashCode để hỗ trợ so sánh đối tượng [12] .

Dart Data Class Plugin cung cấp các tính năng tự động tạo mã giúp tăng năng suất lập trình viên khi làm việc với các lớp dữ liệu. Cụ thể, plugin này có thể sinh mã cho các thành phần như: Named Argument Constructor để khởi tạo đối tượng dễ dàng với các tham số có tên; copyWith để sao chép một thể hiện và ghi đè các giá trị cụ thể mà không làm thay đổi đối tượng gốc; toMap dùng để chuyển đổi đối tượng thành một bản đồ kiểu Map<String, dynamic>, phục vụ cho việc lưu trữ hoặc truyền tải dữ liệu; và fromMap để xây dựng lại đối tượng từ bản đồ dữ liệu đó, hỗ trợ ánh xạ dữ liệu từ JSON hoặc các nguồn tương tự. Nhờ các tính năng này, việc thao tác và quản lý dữ liệu trong các ứng dụng Flutter trở nên nhanh chóng và hiệu quả hơn [11].

2.5.5 Dart Json Serialization Generator

Dart JSON Serialization Generator là một Plugin hỗ trợ tự động sinh mã cho việc ánh xạ dữ liệu giữa đối tượng Dart và dữ liệu JSON, giúp tiết kiệm thời gian và hạn chế lỗi khi lập trình Flutter [13].

Dart JSON Serialization Generator cung cấp nhiều tính năng nổi bật nhằm đơn giản hóa việc xử lý dữ liệu JSON trong các ứng dụng Flutter. Cụ thể, nó sử dụng annotation (chú thích) như @JsonSerializable() để tự động tạo các phương thức như fromJson() để tạo đối tượng từ JSON Map và toJson() để chuyển đối tượng thành JSON Map, giúp tiết kiệm thời gian và tránh lỗi khi viết tay. Công cụ này còn hỗ trợ các đối tượng lồng nhau (nested objects), giá trị mặc định (default value) và tên trường tùy chỉnh (custom field names), đảm bảo linh hoạt và phù hợp với nhiều tình huống khác nhau. Nhờ đó, mã nguồn được giữ sạch sẽ, dễ đọc và dễ bảo trì.

2.6 CÔNG CỤ QUẢN LÝ TRẠNG THÁI

2.6.1 Quản lý trạng thái mặc định của Flutter - setState()

2.6.1.1 Khái niệm

setState() là một phương thức được cung cấp bởi lớp State trong Flutter. Phương thức này thường được sử dụng để cập nhật trạng thái của StatefulWidget. Phương thức này yêu cầu Flutter xây dựng lại widget với trạng thái đã cập nhật. Đây là một cách đơn giản và tích hợp sẵn để quản lý trạng thái của cây widget [14].

2.6.1.2 Ví dụ minh họa

```
onPressed: (context) async {
  await CartSnapshot.deleteCart(item.mon.id, user!.id);
  setState(() {
    selectedItems.remove(item.mon.id);
    dsCart = CartSnapshot.getCart();
  });
  MessDialog.showSnackBar(context, message: "Đã xóa ${item.mon.ten} ra khỏi giỏ hàng");
},
```

Hình 2.3. Ví dụ minh họa sử dụng setState()

Trong đoạn mã trên, setState() được sử dụng để cập nhật giao diện người dùng sau khi một món hàng được xóa khỏi giỏ. Khi người dùng ấn nút xóa, ứng dụng sẽ gọi hàm CartSnapshot.deleteCart() để xóa món hàng tương ứng khỏi Supabase.

Sau khi xóa thành công, phương thức setState() được gọi để cập nhật lại hai giá trị: danh sách các món trong giỏ hàng đang được chọn (selectedItems) và danh sách giỏ hàng hiển thị (dsCart). Cụ thể, ID của món hàng vừa bị xóa sẽ được loại khỏi selectedItems, và dsCart được gán lại từ kết quả mới của CartSnapshot.getCart(), phản ánh danh sách giỏ hàng sau khi món đã bị loại bỏ.

Việc gọi setState() ở đây rất quan trọng, vì nó thông báo cho Flutter biết rằng dữ liệu đã thay đổi và giao diện cần được vẽ lại. Nếu không có setState(), giao diện sẽ không tự động cập nhật, và người dùng sẽ không thấy món hàng vừa bị xóa biến mất khỏi giỏ hàng.

2.6.1.3 Ưu điểm - nhược điểm

Bảng 2.5. Ưu điểm và nhược điểm của setState()

Ưu điểm	Nhược điểm
- Đơn giản, dễ hiểu - Nhanh chóng cập nhật UI: Tự động gọi lại build() sau khi cập nhật, giúp cập nhật giao diện ngay tức thì. - Không cần cài thêm package hay thư viện bên ngoài. - Chỉ widget gọi setState bị rebuild, không ảnh hưởng toàn bộ ứng dụng.	- Hiệu suất: Gọi setState sẽ xây dựng lại toàn bộ tiện ích, điều này có thể không hiệu quả nếu cây widget lớn [14]. - Trạng thái toàn cục: Không phù hợp để quản lý trạng thái ứng dụng toàn cục hoặc chia sẻ trạng thái giữa các phần xa nhau của ứng dụng [14].

2.6.2 Quản lý trạng thái GetX

2.6.2.1 Khái niệm

GetX là một thư viện quản lý trạng thái nhanh, ổn định và nhẹ trong Flutter giúp đơn giản hóa quá trình quản lý và cập nhật trạng thái của ứng dụng [15].

GetX có 3 nguyên lý cốt lõi: Hiệu suất (Productivity), Năng suất (Performance) và Tổ chức (Organization) , đây là các ưu tiên chính trong toàn bộ thiết kế thư viện [16]:

- Hiệu suất: GetX tối ưu hiệu năng và giảm tối đa tài nguyên sử dụng. Không dùng Stream hay ChangeNotifier, giúp ứng dụng phản hồi nhanh và mượt mà.

- Năng suất: GetX sử dụng cú pháp dễ dàng và dễ chịu. Bất kể bạn muốn làm gì, luôn có một cách dễ dàng hơn với GetX. Nó sẽ tiết kiệm hàng giờ phát triển và sẽ cung cấp hiệu suất tối đa mà ứng dụng của bạn có thể cung cấp.
- Tổ chức: GetX tách biệt hoàn toàn giữa giao diện (View), logic trình bày, logic nghiệp vụ, định tuyến và inject phụ thuộc. Không cần BuildContext để điều hướng hay truy cập controller. Không phụ thuộc vào cây widget (InheritedWidget, MultiProvider,...). Quản lý phụ thuộc độc lập với UI giúp tăng khả năng tái sử dụng và bảo trì mã.

GetX có hai trình quản lý trạng thái khác nhau. Một là trình quản lý trạng thái đơn giản (GetBuilder). Hai là trình quản lý trạng thái phản ứng (reactive state manager - GetX/Obx)

Bảng 2.6. Trình quản lý trạng thái của GetX

Trạng thái phản ứng (Reactive State Management)	- Các biến có thể quan sát được (observable) được tạo bằng cách thêm .obs vào giá trị. - Khi giá trị thay đổi, tất cả các widget đang "lắng nghe" biến đó (thông qua Obx) tự động được cập nhật lại.
Trình quản lý trạng thái đơn giản	- Dùng GetxController để chứa logic xử lý và dữ liệu. - Widget dùng GetBuilder sẽ rebuild lại khi update() được gọi trong controller.

2.6.2.2 Ví dụ minh họa

```dart
//search_controller
class lichSuTimKiem extends GetxController {
  var dstk = <String>[].obs;
  var dsDeXuat = <Mon>[].obs;
  var dsDeXuatFilter = <Mon>[].obs;
  TextEditingController txtSearch = TextEditingController();

  @override
  void onInit() {
    super.onInit();
    layMenu();
    txtSearch.addListener(() {
      timKiemTheoTen(txtSearch.text);
    });
  }

  void layMenu() async {
    List<Mon> allItems = await ChangTeaSnapshot.getMon();
    dsDeXuat.value = allItems;
    dsDeXuatFilter.value = allItems;
  }

  void addTimKiem(String txt) {
    if (!dstk.contains(txt)) {
      dstk.add(txt);
    }
  }

  void timKiemTheoTen(String txt) {
    final tuKhoa = txt.toLowerCase().trim();
    dsDeXuatFilter.value = dsDeXuat.where(
            (item) => item.ten.toLowerCase().contains(tuKhoa)
    ).toList();
  }
}
```

```dart
//lichSuTimKiem ds = Get.put(lichSuTimKiem());
Expanded(
  child: Obx(() {
    if (txtSearch.text.isEmpty) {
      return ListView.separated(
        itemBuilder: (context, index) {
          // Hiển thị kết quả tìm kiếm
          return Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(ds.dstk[index], style: TextStyle(fontSize: 15, fontStyle: FontStyle.italic),
            ),  // Text
          );  // Padding
        },
        separatorBuilder: (context, index) => Divider(),
        itemCount: ds.dstk.length,
      );  // ListView.separated
    }
    return ListView.separated(
      itemBuilder: (context, index) {
        final item = ds.dsDeXuatFilter[index];
        return InkWell(
          onTap: () {
            Navigator.push(context, MaterialPageRoute(builder: (context) => ChitietItem(menu_changtea: item),),);
          },
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(item.ten, style: TextStyle(fontSize: 15, fontStyle: FontStyle.italic)),
          ),  // Padding
        );  // InkWell
      },
      separatorBuilder: (context, index) => Divider(),
      itemCount: ds.dsDeXuatFilter.length,
    );},  // ListView.separated
  ),  // Obx
)  // Expanded
```

Hình 2.4. Ví dụ minh họa sử dụng GetX

Trong đoạn mã trên, GetX được sử dụng để quản lý trạng thái cho tính năng tìm kiếm món. Lớp lichSuTimKiem kế thừa từ GetxController, lớp này chứa ba biến danh sách (dstk, dsDeXuat, dsDeXuatFilter) được khai báo với .obs cho phép GetX theo dõi sự thay đổi giá trị để cập nhật giao diện tương ứng.

Tại phương thức onInit(), controller gọi hàm layMenu() để lấy toàn bộ danh sách món từ cơ sở dữ liệu và đồng thời lắng nghe thay đổi từ TextEditingController của ô tìm kiếm (txtSearch). Mỗi khi người dùng nhập liệu, hàm timKiemTheoTen() được gọi để lọc lại danh sách món theo từ khóa, kết quả được gán vào dsDeXuatFilter, từ đó kích hoạt cơ chế cập nhật lại giao diện thông qua Obx().

Ở phần giao diện, Obx() hiển thị danh sách tìm kiếm. Khi người dùng chưa nhập gì, giao diện sẽ hiển thị dstk (lịch sử tìm kiếm). Khi có từ khóa, dsDeXuatFilter sẽ được sử dụng để hiển thị danh sách gợi ý. Nhờ vào cơ chế reactive của GetX, bất kỳ thay đổi nào trên dstk hoặc dsDeXuatFilter đều tự động cập nhật lại giao diện mà không cần gọi setState().

2.6.2.3 Ưu điểm – nhược điểm

Bảng 2.7. Ưu điểm – nhược điểm của GetX

Ưu điểm	Nhược điểm

- Nhẹ và hiệu quả: GetX cực kỳ nhẹ và hiệu quả, lý tưởng cho các ứng dụng quy mô nhỏ đến lớn. Nó cung cấp phương pháp tiếp cận phản ứng với quản lý trạng thái chỉ xây dựng lại các phần của giao diện người dùng cần cập nhật, đảm bảo ứng dụng của bạn chạy trơn tru và hiệu quả [15]. - Dễ học: GetX có đường cong học tập nhỏ, giúp các nhà phát triển dễ dàng học và triển khai. Nó cung cấp các API trực quan, dễ hiểu và dễ sử dụng, khiến nó trở thành lựa chọn tuyệt vời cho cả nhà phát triển mới bắt đầu và có kinh nghiệm [15]. - Tích hợp dependency injection: GetX bao gồm dependency injection tích hợp, cho phép bạn dễ dàng quản lý các phụ thuộc trong ứng dụng của mình. Tính năng này giúp bạn dễ dàng chuyển đổi giữa các phụ thuộc và quản lý chúng trên toàn bộ ứng dụng của mình [15].	- Phụ thuộc vào thư viện: GetX là một thư viện bên thứ ba, nên ứng dụng sẽ phụ thuộc vào nó. Nếu thư viện không được duy trì hoặc có lỗi trong các bản cập nhật, dự án có thể gặp rủi ro. - Không phù hợp với mọi dự án: Trong các dự án lớn hoặc phức tạp, GetX có thể không đủ mạnh để quản lý trạng thái phức tạp.

Chương 3. PHÂN TÍCH HỆ THỐNG

3.1 KHẢO SÁT – ĐÁNH GIÁ THỰC TRẠNG

3.1.1 Khảo sát thực trạng

Quán trà sữa ChangTea hoạt động từ 8h30 đến 22h00 hàng ngày kể cả ngày lễ, phục vụ đa dạng đối tượng khách hàng từ học sinh, sinh viên đến người trưởng thành. Quán cung cấp thực đơn phong phú với nhiều loại đồ uống và đồ ăn phù hợp với khẩu vị Gen Z, giá dao động từ 23.000đ đến 40.000đ. Điểm khác biệt chính của ChangTea là sử dụng các loại trà cao cấp nhập khẩu. Hiện tại, ChangTea đang phục vụ khách hàng qua nhiều kênh: trực tiếp tại quán, qua số hotline, Facebook.

Quán thường xuyên tổ chức các chương trình khuyến mãi như Free Upsize, mua 2 tặng 1, và hệ thống tích điểm, nhưng quy trình này đang được quản lý thủ công. ChangTea có lượng theo dõi gần 1.500 người trên Facebook với mức đánh giá cao (4.7/5 sao) trên Google Map.

Tuy nhiên, quán đang gặp nhiều khó khăn trong việc quản lý đơn hàng, đặc biệt là vào giờ cao điểm với chỉ 2-3 nhân viên mỗi ca làm việc. Quán vẫn đang sử dụng phương pháp ghi giấy để đặt và gọi món, dẫn đến việc quản lý thứ tự đơn hàng không nhất quán, đồng thời không có hệ thống lưu trữ lịch sử mua hàng của khách để hỗ trợ cho các chương trình tích điểm và chăm sóc khách hàng.

3.1.2 Đánh giá thực trạng

Qua khảo sát thực tế, ChangTea đang vận hành mà không có ứng dụng mobile riêng cho khách hàng đặt hàng. Hiện tại, khách hàng chỉ có thể đặt hàng trực tiếp tại quán, qua điện thoại, Facebook. Việc thiếu một ứng dụng mobile độc lập dẫn đến nhiều hạn chế trong quy trình kinh doanh.

Thứ nhất, quán đang sử dụng phương pháp thủ công để ghi nhận đơn hàng, dẫn đến nhiều sai sót và chậm trễ trong phục vụ, đặc biệt vào giờ cao điểm. Thứ hai, thực đơn trên các nền tảng trực tuyến không đồng bộ với thực đơn tại quán, khiến khách hàng bị nhầm lẫn và thất vọng khi không tìm thấy sản phẩm yêu thích. Thứ ba, không có hệ thống hiển thị các món best seller hoặc đề xuất sản phẩm dựa trên lịch sử mua hàng, làm giảm cơ hội bán hàng chéo và nâng cấp đơn hàng. Quán cũng không có cơ chế tìm kiếm nhanh chóng để khách hàng dễ dàng tìm kiếm sản phẩm theo nhu cầu. Hệ thống giỏ

hàng và hóa đơn hiện tại hoàn toàn thủ công, tạo ra khả năng sai sót cao và không minh bạch trong việc tính toán chi phí. Chương trình khuyến mãi và tích điểm đang được triển khai nhưng không có nền tảng số hóa để quản lý hiệu quả, khiến việc theo dõi trở nên phức tạp và nhiều sai sót. Ngoài ra, thiếu ứng dụng mobile còn khiến quán mất đi một kênh marketing tiềm năng để kết nối trực tiếp với khách hàng, thông báo về sản phẩm mới và chương trình khuyến mãi. Trong bối cảnh thị trường trà sữa cạnh tranh gay gắt với nhiều đối thủ đã áp dụng công nghệ đặt hàng hiện đại, việc ChangTea chưa có ứng dụng mobile riêng đang dần trở thành điểm yếu cạnh tranh đáng kể.

3.1.3 Đề xuất hướng phát triển hệ thống

Dựa trên đánh giá hiện trạng, việc phát triển ứng dụng mobile đặt hàng cho ChangTea là giải pháp cấp thiết để nâng cao hiệu quả hoạt động và trải nghiệm khách hàng. Ứng dụng cần tập trung vào các chức năng cốt lõi phục vụ quá trình đặt hàng đơn giản và hiệu quả.

Đầu tiên, ứng dụng cần có giao diện hiển thị sản phẩm rõ ràng, trực quan với hình ảnh chất lượng, mô tả chi tiết về thành phần, giá cả và tùy chọn (size, đá, đường, topping). Chức năng tìm kiếm thông minh cần được tích hợp để khách hàng dễ dàng tìm kiếm sản phẩm theo tên, loại hoặc thành phần. Một phần quan trọng khác là khả năng hiển thị các món best seller của quán, giúp khách hàng mới dễ dàng chọn lựa các sản phẩm được ưa chuộng. Hệ thống giỏ hàng trực quan cho khách hàng thêm, sửa, xóa sản phẩm một cách dễ dàng, đồng thời hiển thị rõ tổng tiền. Khi xác nhận đặt hàng, ứng dụng sẽ tạo hóa đơn chi tiết, bao gồm danh sách sản phẩm, giá tiền, và các thông tin giao hàng (nếu có).

Việc chuyển đổi từ hệ thống đặt hàng thủ công sang ứng dụng mobile sẽ giúp ChangTea tối ưu hóa quy trình phục vụ, giảm thiểu sai sót, nâng cao trải nghiệm khách hàng và tạo ra kênh bán hàng mới hiệu quả, góp phần củng cố thương hiệu trong thị trường trà sữa ngày càng cạnh tranh.

3.2 MÔ TẢ ỨNG DỤNG ĐẶT TRÀ SỮA CHANGTEA

Ứng dụng đặt trà sữa ChangTea mang đến trải nghiệm tiện lợi và nhanh chóng cho khách hàng khi tiếp cận các món thức uống và món ăn đặc sắc của quán. Ứng dụng có giao diện thân thiện giúp người dùng dễ xem menu với hình ảnh cụ thể, giá cả rõ ràng

và các thông tin của món. Bên cạnh đó, ứng dụng còn cho phép khách hàng tìm kiếm món theo tên hoặc đề xuất các món tương tự dựa trên sở thích. Người dùng có thể chọn món và thêm vào giỏ hàng, điều chỉnh số lượng hoặc xoá nếu cần. Sau khi đặt hàng, khách hàng có thể xem chi tiết hóa đơn và theo dõi lịch sử các đơn hàng đã mua. Ứng dụng cũng hỗ trợ quản lý thông tin tài khoản cá nhân như tên, địa chỉ, số điện thoại,... giúp việc đặt hàng nhanh chóng và thuận tiện hơn trong những lần sau.

3.3 CÁC YÊU CẦU, CHỨC NĂNG CƠ BẢN

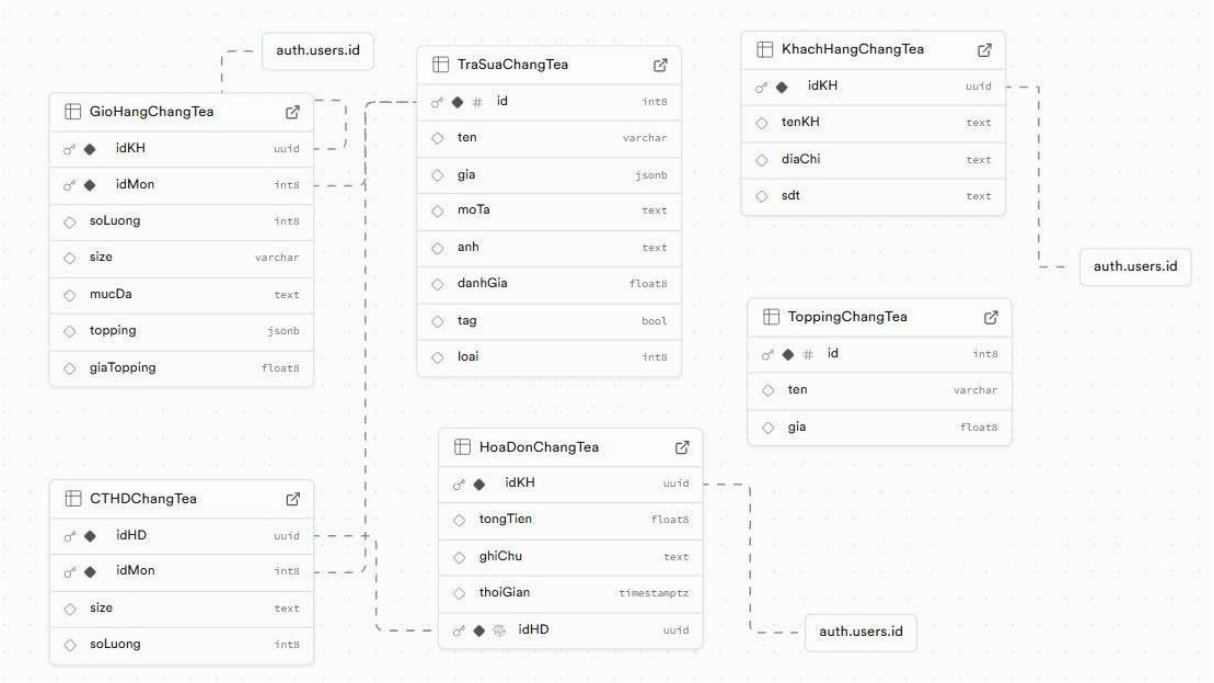
Ứng dụng đặt trà sữa ChangTea bao gồm các chức năng sau:

- Hiển thị danh sách các món trà sữa có hình ảnh và giá, cho phép xem chi tiết.
- Cho phép tìm kiếm món hoặc đề xuất món tương tự.
- Giỏ hàng lưu các món đã chọn ứng với mỗi tài khoản của khách hàng, chỉnh sửa số lượng hoặc xoá món.
- Cho phép xem hóa đơn đặt hàng, chi tiết hóa đơn.
- Cho phép đăng nhập để mua hàng, hiển thị các thông tin của khách hàng.
- Lưu và hiển thị lịch sử mua.

Chương 4. THIẾT KẾ HỆ THỐNG

4.1 CƠ SỞ DỮ LIỆU

4.1.1 Lược đồ cơ sở dữ liệu



Hình 4.1. Lược đồ CSDL của ứng dụng đặt trà sữa ChangTea

4.1.2 Mối quan hệ giữa các bảng

Bảng 4.1. Mối quan hệ giữa các bảng với nhau

Bảng	Liên kết với	Mối quan hệ	Mô tả
auth.users	KhachHangChangTea	1 – 1	Một user ứng với một khách hàng
KhachHangChangTea	GioHangChangTea	1 - 1	Mỗi khách hàng có một giỏ hàng
KhachHangChangTea	HoaDonChangTea	1 - N	Một khách có thể có nhiều hóa đơn
TraSuaChangTea	GioHangChangTea	1 - N	Một món có thể nằm trong nhiều giỏ hàng

TraSuaChangTea	CTHDChangTea	1 - N	Một món có thể xuất hiện trong nhiều chi tiết hóa đơn
HoaDonChangTea	CTHDChangTea	1 - N	Mỗi hóa đơn gồm nhiều chi tiết hóa đơn

4.1.3 Mô tả chi tiết các bảng

Bảng 4.2. Trà sữa ChangTea

Thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int8	Mã món	Khóa chính
ten	varchar	Tên món	
gia	jsonb	Giá của món ứng với size	
moTa	text	Mô tả cụ thể món	
anh	text	Ảnh của món	
danhGia	float8	Đánh giá của món	
tag	bool	True ứng với Best Seller	
loai	int8	1 là Trà Sữa, 2 là Trà Trái Cây, 3 là Đồ Ăn	

Bảng 4.3. Topping ChangTea

Thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int8	Mã topping	Khóa chính
ten	varchar	Tên topping	
gia	float8	Giá topping	

Bảng 4.4. Khách hàng ChangTea

Thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
idKH	uuid	Mã khách hàng	Khóa chính Khóa ngoại bảng users (Supabase)
tenKH	text	Tên khách hàng	
diaChi	text	Địa chỉ của khách hàng	
sdt	text	Số điện thoại của khách hàng	

Bảng 4.5. Giỏ hàng ChangTea

Thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
idKH	uuid	Mã khách hàng	Khóa chính Khóa ngoại bảng users (Supabase)
idMon	int8	Mã món	Khóa chính Khóa ngoại bảng TraSuaChangTea
soLuong	int8	Số lượng món	
size	varchar	kích thước món	
mucDa	text	Mức đá của món	
topping	json	Tên topping ứng với giá	
giaTopping	float8	Tổng giá các topping của món	

Bảng 4.6. Hóa Đơn ChangTea

Thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
idKH	uuid	Mã khách hàng	Khóa chính Khóa ngoại bảng users (Supabase)
idHD	uuid	Mã hóa đơn	Khóa chính (Tự sinh)
thoiGian	timestamptz	Thời gian đặt đơn hàng	
tongTien	float8	Tổng tiền các món	
ghiChu	text	Các ghi chú của khách hàng	

Bảng 4.7. Chi Tiết Hóa Đơn ChangTea

Thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
idHD	uuid	Mã hóa đơn	Khóa chính Khóa ngoại bảng HoaDonChangTea
idMon	int8	Mã món	Khóa chính Khóa ngoại bảng TraSuaChangTea
soLuong	int8	Số lương món	
size	text	Kích thước món	

4.2 THIẾT KẾ MODEL VÀ CONTROLLER

4.2.1 Trang chủ (Home)

Giao diện Trang chủ được xây dựng qua hàm buildHome, chứa các thành phần như slide ảnh quảng cáo, danh mục món (trà sữa, trà trái cây, ăn vặt), và liên kết điều hướng đến các trang chi tiết tương ứng.

Dữ liệu hiển thị lấy từ tầng Model Mon, được lọc theo thuộc tính loai để phân loại các nhóm thực đơn.

Ngoài ra, hàm slide_Anh() được sử dụng như một controller function, giúp tự động chuyển ảnh sau mỗi 3 giây, tạo hiệu ứng trình chiếu liên tục.

```
void slide_Anh({
  required PageController p,
  required int Function() getCurrentPage,
  required int totalPages,
  required void Function(int newPage) onPageChanged,
  required void Function(Timer timer) onTimerCreated,
})
{
  Timer t = Timer.periodic(
    const Duration(seconds: 3), (Timer timer) {
    int currentPage = getCurrentPage();
    int nextPage = (currentPage + 1) % totalPages;

    p.animateToPage(
      nextPage,
      duration: const Duration(milliseconds: 500),
      curve: Curves.easeInOut,
    );
    onPageChanged(nextPage);
    }
  );  // Timer.periodic
  onTimerCreated(t);
}
```

Hình 4.2 Hàm slide_Anh()

Ngoài ra còn có class MessDialog là một UX Controller dùng để xử lý các hành động như thông báo, xác nhận thoát, hoặc hiện thông báo dạng snack bar.

```dart
class MessDialog {
  static Future<bool> wannaOut(BuildContext context) async {
    final shouldExit = await showDialog<bool>(
      context: context,
      builder: (context) => AlertDialog(
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
        title: const Center(
          child: Text(
            'Xác nhận thoát',
            style: TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 20,
              color: Colors.redAccent,
            ),  // TextStyle
          ),  // Text
        ),  // Center
        content: const Text(
          'Bạn có chắc chắn muốn thoát ứng dụng không ?',
          textAlign: TextAlign.center,
          style: TextStyle(fontSize: 16),
        ),  // Text
        actionsPadding: const EdgeInsets.only(bottom: 16),
        actionsAlignment: MainAxisAlignment.center,
        actions: [
          ElevatedButton(
```

Hình 4.3. Minh họa class MessDialog

4.2.2 Menu

4.2.2.1 Class Mon

Lớp Mon đại diện cho một đơn vị thức uống cụ thể (trà sữa, trà trái cây...), tương ứng với bảng TraSuaChangTea trên Supabase. Lớp này chứa đầy đủ thông tin của một món như tên, mô tả, hình ảnh, loại sản phẩm, và đặc biệt là giá bán theo từng size, cụ thể:

```dart
class Mon {
  int id;
  String ten;
  Map<String, double> gia;
  String? moTa;
  String? anh;
  bool tag;
  double danhGia;
  int loai;

  Mon({
    required this.id,
    required this.ten,
    required this.gia,
    this.moTa,
    this.anh,
    required this.tag,
    required this.danhGia,
    required this.loai,
  });
```

Hình 4.4. Các thuộc tính của lớp Món

- id (int): Mã định danh duy nhất cho từng món, đảm bảo tính duy nhất khi xử lý dữ liệu.
- ten (String): Tên của món được hiển thị trong danh sách sản phẩm.
- gia (Map<String, double>): sử dụng Map để lưu trữ giá tương ứng với từng size (ví dụ: S: 25.000, M: 30.000) → giúp dễ dàng hiển thị và tính toán giá theo lựa chọn của người dùng.
- moTa (String?): Mô tả chi tiết về món (tuỳ chọn).
- anh (String?): Đường dẫn ảnh món, giúp hiển thị trực quan trong UI.
- tag (bool): Cờ boolean để đánh dấu các món "best seller".
- danhGia (double): Điểm đánh giá từ người dùng.
- loai (int): Dùng để phân loại món (trà sữa/trà trái cây/ăn vặt).

Các hàm tạo và JSON:

```
  factory Mon.fromJson(Map<String, dynamic> json) {
    return Mon(
      id: json["id"],
      ten: json["ten"],
      gia: (json['gia'] as Map<String, dynamic>)
          .map((key, value) => MapEntry(key, (value as num).toDouble())),
      moTa: json["moTa"],
      anh: json["anh"],
      tag: json["tag"],
      danhGia: (json["danhGia"] as num).toDouble(),
      loai: json["loai"],
    );  // Mon
  }

  Map<String, dynamic> toJson() {
    return {
      "id": id,
      "ten": ten,
      "gia": gia,
      "moTa": moTa,
      "anh": anh,
      "tag": tag,
      "danhGia": danhGia,
      "loai": loai
    };
  }
}
```

Hình 4.5. Hàm chuyển đổi dữ liệu Món

- factory Mon.fromJson(): Chuyển dữ liệu từ Supabase (dạng Map<String, dynamic>) sang đối tượng Mon.
- toJson(): Chuyển ngược lại đối tượng Mon thành Map để gửi lên server nếu cần.

4.2.2.2 Class Topping

Class Topping tương ứng với bảng ToppingChangTea trong cơ sở dữ liệu, thể hiện các thành phần phụ đi kèm món uống như trân châu, thạch, phô mai...

- Các thuộc tính:

- id (int): Mã topping.
- ten (String): Tên topping hiển thị.
- gia (double): Giá tiền của topping, được cộng vào đơn hàng nếu người dùng chọn.

- Phương thức chính: Topping.fromJson(Map<String, dynamic> json): Dùng để chuyển dữ liệu từ Supabase sang đối tượng Topping.

- Topping là phần mở rộng của món chính, được tính phí thêm. Mỗi topping sẽ được dùng để tính tổng giá cuối cùng khi người dùng chọn.

```dart
class Topping{
  int id;
  String ten;
  double gia;

  Topping({
    required this.id,
    required this.ten,
    required this.gia,
  });

  factory Topping.fromJson(Map<String, dynamic> json) {
    return Topping(
      id: json["id"],
      ten: json["ten"],
      gia: (json["gia"] as num).toDouble()
    );
  }
}
```

Hình 4.6. Class Topping

4.2.2.3 Class ChangTeaSnapshot

ChangTeaSnapshot là một class trung gian chứa các hàm gọi đến Supabase để thao tác dữ liệu của Mon và Topping.

Các phương thức chính bao gồm:

- Future<List<Mon>> getMon(): Lấy danh sách toàn bộ món uống từ bảng TraSuaChangTea, ánh xạ thành danh sách đối tượng Mon.
- Future<List<Topping>> getTopping(): Lấy danh sách topping từ bảng ToppingChangTea, ánh xạ thành danh sách Topping.
- double tinhTongGiaTopping(List<Topping> toppings): Nhận vào danh sách topping người dùng đã chọn, trả về tổng chi phí các topping đó.
- void addTopping(Topping t, List<Topping> ds): Thêm topping vào danh sách nếu chưa tồn tại.

```dart
class ChangTeaSnapshot{
  static Future <List<Mon>> getMon() async{
    final supabase = Supabase.instance.client;
    List<Mon> dsMon = [];
    final data = await supabase.from('TraSuaChangTea').select();
    dsMon = data.map((e) => Mon.fromJson(e),).toList();
    return dsMon;
  }

  static Future <List<Topping>> getTopping() async{
    final supabase = Supabase.instance.client;
    List<Topping> dsTopping = [];
    final data = await supabase.from('ToppingChangTea').select();
    dsTopping = data.map((e) => Topping.fromJson(e),).toList();
    return dsTopping;
  }

  static double tinhTongGiaTopping(List<Topping> toppings) {
    double tong = 0;
    for(var i in toppings)
      tong += i.gia;
    return tong;
  }

  static void addTopping(Topping t, List<Topping> ds){
    if(ds.contains(t.id)){
      ds.add(t);
    }
  }
}
```

Hình 4.7. Class ChangTeaSnapshot

4.2.3 Giỏ hàng (Cart)

4.2.3.1 Cart Model

a. CartItem

CartItem là lớp đại diện cho một mục trong giỏ hàng, mô hình hóa dữ liệu từ bảng GioHangChangTea trong cơ sở dữ liệu Supabase. Mỗi mục giỏ hàng gồm nhiều thuộc tính phản ánh chi tiết một món được thêm vào: mon, soLuong, size, mucDa, topping, giaTopping.

```dart
class CartItem {
  final Mon mon;
  final int soLuong;
  final String size;
  final String mucDa;
  final String topping;
  final double giaTopping;

  CartItem({
    required this.mon,
    required this.soLuong,
    required this.size,
    required this.mucDa,
    required this.topping,
    required this.giaTopping,
  });

  factory CartItem.fromMap(Map<String, dynamic> map){
    return CartItem(
      mon: Mon.fromJson(map['TraSuaChangTea']),
      soLuong: map['soLuong'] as int,
      size: map['size'],
      mucDa: map['mucDa'],
      topping: map['topping'],
      giaTopping:  (map['giaTopping'] as num).toDouble(),
    );  // CartItem
  }
}
```

Hình 4.8. Code model CartItem

Dùng hàm tạo từ Map (dạng Map<String, dynamic>) để ánh xạ dữ liệu từ Supabase thành đối tượng CartItem.

Bảng 4.8. Giải thích code của hàm Map

Dòng	Giải thích
mon: Mon.fromJson(map['TraSuaChangTea'])	Dữ liệu từ bảng TraSuaChangTea được ánh xạ sang đối tượng Mon nhờ hàm fromJson() có sẵn trong lớp Mon.
soLuong: map['soLuong'] as int	Lấy số lượng món trong giỏ hàng từ key ‘soLuong’, ép kiểu sang int.

size: map['size']	Lấy giá trị kích thước món uống từ key 'size', dạng String.
mucDa: map['mucDa']	Lấy mức đá của món, dạng String.
topping: map['topping']	Lấy loại topping, là một chuỗi mô tả các topping người dùng chọn.
giaTopping: (map['giaTopping'] as num).toDouble()	Giá topping được lưu dưới dạng num, nên cần ép sang double để sử dụng trong các phép tính tiền.

b. CartSnapshot

- CartSnapshot là lớp tiện ích chứa các hàm giúp tương tác trực tiếp với Supabase, gồm: getCart(), insertCart(), deleteCart().

+ Lấy dữ liệu giỏ hàng (getCart()).

```
34  class CartSnapshot {
35    static Future<List<CartItem>> getCart() async{
36      final user = supabase.auth.currentUser;
37      var response = await supabase.from('GioHangChangTea').select
         ("soLuong, size, mucDa, topping, giaTopping, TraSuaChangTea(*)")
         .eq("idKH", user!.id);
38      return response.map((e) => CartItem.fromMap(e),).toList();
39    }
```

Hình 4.9. Code hàm getCart()

- Truy vấn Supabase để lấy toàn bộ giỏ hàng (GioHangChangTea) của người dùng hiện tại.
- Dùng .select(...) để lấy:

+ Thông tin đơn hàng: soLuong, size, mucDa, topping, giaTopping.

+ TraSuaChangTea(*): lấy chi tiết món từ bảng TraSuaChangTea thông qua khóa ngoại (foreign key).

+ .eq("idKH", user.id) để đảm bảo chỉ lấy đúng dữ liệu của người dùng đang đăng nhập.

- Sau đó, dùng CartItem.fromMap() để chuyển đổi từng dòng dữ liệu thành đối tượng CartItem.

→ Hiển thị dữ liệu giỏ hàng lên giao diện ứng dụng, đảm bảo người dùng thấy được toàn bộ món đã chọn.

+ ***Thêm món vào giỏ hàng (insertCart()):*** thêm một món mới vào bảng GioHangChangTea trong Supabase.

```
41    static Future<void> insertCart(CartItem c) async{
42      final user = supabase.auth.currentUser;
43      await supabase.from('GioHangChangTea').insert({
44        "idKH": user!.id,
45        "idMon": c.mon.id,
46        "soLuong": c.soLuong,
47        "size": c.size,
48        "mucDa": c.mucDa,
49        "topping": c.topping,
50        "giaTopping": c.giaTopping,
51      });
52    }
53
```

Hình 4.10.Code hàm insertCart()

- Lấy user.id từ supabase.auth.
- Gọi .insert({...}) để chèn bản ghi mới vào bảng GioHangChangTea.
- Các trường được chèn tương ứng với các thuộc tính trong đối tượng CartItem.

→ Khi người dùng chọn món từ menu và thêm vào giỏ, dữ liệu sẽ được lưu trữ lại trên Supabase để duy trì trạng thái.

+ ***Xóa món khỏi giỏ hàng (deleteCart()):*** xóa một món cụ thể khỏi giỏ hàng của người dùng (trong trường hợp đổi ý hoặc chọn nhầm,…).

```
54      static Future<void> deleteCart(int idMon, String idKH) async {
55        final user = supabase.auth.currentUser;
56        final idKH = user?.id;
57
58        await supabase.from('GioHangChangTea').delete().eq('idMon',
           idMon).eq('idKH', idKH!);
59      }
60    }
```

Hình 4.11. Code hàm deleteCart()

- Truy cập bảng GioHangChangTea.
- Dùng .delete() kết hợp với .eq() để xoá bản ghi thỏa 2 điều kiện:
  - idMon: ID của món muốn xoá.
  - idKH: ID người dùng hiện tại.

4.2.3.2 CartController

CartController dùng để điều khiển logic giỏ hàng: quản lý danh sách món có trong giỏ, tính tổng tiền, xử lý chọn hoặc xóa món, giao tiếp với Supabase thông qua CartSnapshot.

```
class CartController extends GetxController {
  var cartItems = <String, CartItem>{}.obs;
  var selectedItems = <String, bool>{}.obs;
  var isAllSelected = false.obs;
  var cartList = <CartItem>[].obs;
```

Hình 4.12. Các biến của CartController

Bảng 4.9. Danh sách các biến dùng trong CartController

Biến	Kiểu	Mô tả
cartItems	Map<String, CartItem>.obs	Các món trong giỏ, key là ID hoặc hash
selectedItems	Map<String, bool>.obs	Món đang được chọn
isAllSelected	RxBool	Trạng thái chọn tất cả

cartList	RxList<CartItem>	Danh sách giỏ hàng lấy từ Supabase

→ Các biến reactive trong GetX, giúp giao diện cập nhật ngay khi có thay đổi.

- Hàm fetchCart()

```
Future<void> fetchCart() async {
  if (user == null) return;
  final result = await CartSnapshot.getCart();
  cartList.value = result;
}

@override
void onInit() {
  super.onInit();
  fetchCart();
}
```

Hình 4.13. Hàm fetchCart()

Hàm fetchCart() là một hàm bất đồng bộ có nhiệm vụ tải dữ liệu giỏ hàng của người dùng từ cơ sở dữ liệu Supabase thông qua model CartSnapshot. Khi được gọi, nó kiểm tra xem người dùng đã đăng nhập chưa (user == null), sau đó gọi CartSnapshot.getCart() để truy vấn toàn bộ các món có trong giỏ hàng. Kết quả được gán vào biến cartList, một RxList để đảm bảo khi dữ liệu thay đổi thì giao diện cũng được cập nhật tự động. Hàm này thường được gọi trong onInit() để tải dữ liệu ngay khi controller khởi tạo.

- Hàm tongCart():

```dart
static CartController get() => Get.find();

Future<int> tongCart(String id) async {
  final user = Supabase.instance.client.auth.currentUser;
  if (user == null) return 0;

  final response = await Supabase.instance.client
      .from('GioHangChangTea')
      .select('soLuong')
      .eq('idKH', user!.id);

  final data = response as List<dynamic>;
  int total = 0;

  for (var item in data) {
    // total += item['soLuong'] as int;
    total += 1 as int;
  }

  return total;
}
```

Hình 4.14. Hàm tongCart()

Hàm tongCart(String id) dùng để tính tổng số dòng (món hàng) trong giỏ của người dùng. Hàm này gửi truy vấn đến bảng GioHangChangTea của Supabase, lọc theo idKH là ID người dùng. Dữ liệu trả về là danh sách số lượng (soLuong) của từng món, nhưng trong đoạn code, số lượng không được cộng dồn mà chỉ đếm số phần tử có (total += 1). Đây là một cách đếm số loại sản phẩm, không phải số lượng thực tế. Kết quả này thường được dùng để hiển thị badge số lượng trên biểu tượng giỏ hàng.

- Hàm tinhTongTien():

```dart
// Tổng tiền các mặt hàng đã chọn trong Giỏ hàng
double tinhTongTien(Map choMH) {
  double tong = 0;
  choMH.forEach((key, isSelected) {
    if (isSelected && cartItems.containsKey(key)) {
      final item = cartItems[key]!;
      final giaSize = item.mon.gia[item.size] ?? 0;
      tong += (giaSize + item.giaTopping) * item.soLuong;
    }
  });
  return tong;
}
```

Hình 4.15. Hàm tinhTongTien()

Hàm tinhTongTien() có nhiệm vụ tính tổng số tiền của các mặt hàng đã được người dùng chọn trong giao diện giỏ hàng. Hàm nhận vào một Map chứa thông tin về các món đã được chọn (true/false), sau đó duyệt từng phần tử. Nếu món đó đang được chọn và có trong cartItems, hệ thống sẽ lấy giá theo size (mon.gia[size]) cộng với giaTopping, nhân với số lượng (soLuong) để tính tổng. Hàm này hỗ trợ tính toán động và hiển thị số tiền tạm tính cho người dùng.

- Hàm soLuongDaChon():

```
int soLuongDaChon(Map selectedItems) {
  int count = 0;
  selectedItems.forEach((key, isSelected) {
    if (isSelected) count++;
  });
  return count;
}
```

Hình 4.16. Hàm soLuongDaChon()

Hàm soLuongDaChon() có chức năng đếm xem có bao nhiêu món trong giỏ hàng đã được người dùng chọn (thông qua checkbox). Hàm duyệt qua từng phần tử và đếm số phần tử có giá trị true. Kết quả dùng để điều khiển các hành động hàng loạt, như "xoá tất cả món đã chọn" hoặc "thanh toán các món đã chọn".

- Hàm chonTatCa():

```
void chonTatCa(bool value) {
  isAllSelected.value = value;
  for (var key in cartItems.keys) {
    selectedItems[key] = value;
  }
}
```

Hình 4.17. Hàm chonTatCa()

Hàm này cho phép người dùng chọn hoặc bỏ chọn tất cả các món hàng trong giỏ chỉ với một thao tác. Hàm nhận vào một bool value, sau đó cập nhật biến isAllSelected để phản ánh trạng thái chọn tất cả. Tiếp theo, hàm duyệt qua tất cả cartItems và gán giá trị value cho mỗi phần tử trong selectedItems.

- Hàm auth():

```
void auth(){
  update(['user_infor']);
}
```

Hình 4.18. Hàm auth()

Hàm auth() chỉ đơn giản gọi update(['user_infor']) để thông báo cho GetX rằng nhóm ID "user_infor" cần được cập nhật, được dùng khi thông tin đăng nhập người dùng thay đổi và cần cập nhật lại giao diện hoặc các phần dữ liệu liên quan. Nó đóng vai trò hỗ trợ tương tác với giao diện phụ thuộc vào trạng thái người dùng.

- Hàm removeItem():

```
  void removeItem(CartItem i) {
    final user = Supabase.instance.client.auth.currentUser;
    CartSnapshot.deleteCart(i.mon.id, user!.id);
    update();
  }
}
```

Hình 4.19. Hàm removeItem()

Hàm removeItem() được sử dụng để xoá một món hàng ra khỏi giỏ. Nó nhận vào một đối tượng CartItem, sau đó gọi đến hàm deleteCart() trong model CartSnapshot, truyền vào ID món hàng (i.mon.id) và ID người dùng. Sau khi thực hiện xoá, hàm gọi update() để cập nhật lại giao diện giỏ hàng. Điều này đảm bảo sau khi xoá, UI sẽ phản ánh ngay thay đổi mà không cần phải làm mới thủ công.

4.2.4 Hóa đơn (Invoice)

4.2.4.1 Invoice Model

Trong hệ thống, dữ liệu hóa đơn được mô hình hóa thông qua hai class chính: HoaDon và CTHD.

a. Class HoaDon

Class HoaDon đại diện cho một hóa đơn, bao gồm các thuộc tính:

- idHD (String?): mã hóa đơn (có thể rỗng vì Supabase sẽ tự sinh).
- tongTien (double): tổng số tiền của hóa đơn.
- thoiGian (DateTime): thời điểm tạo hóa đơn.
- ghiChu (String): ghi chú từ người dùng.

Lớp này có hai phương thức:

- fromMap(): chuyển đổi dữ liệu từ Supabase (dạng Map) thành đối tượng HoaDon.
- toMap(userId): chuẩn bị dữ liệu dưới dạng Map để gửi lên Supabase, kèm theo ID người dùng.

```dart
class HoaDon {
  final String? idHD; // nullable vì Supabase sẽ tự sinh
  final double tongTien;
  final DateTime thoiGian;
  final String ghiChu;

  HoaDon({
    this.idHD, // bỏ không cần truyền nếu tự sinh
    required this.tongTien,
    required this.thoiGian,
    required this.ghiChu,
  });

  factory HoaDon.fromMap(Map<String, dynamic> map) {
    return HoaDon(
      idHD: map['idHD'],
      tongTien: (map['tongTien'] as num).toDouble(),
      thoiGian: DateTime.parse(map['thoiGian']),
      ghiChu: map['ghiChu'] ?? "",
    );
  }

  Map<String, dynamic> toMap(String userId) {
    return {
      "idKH": userId,
      "tongTien": tongTien,
      "thoiGian": thoiGian.toIso8601String(),
      "ghiChu": ghiChu,
    };
  }
}
```

Hình 4.20. Class HoaDon

b. Class CTHD

Class CTHD mô hình hóa từng món trong hóa đơn, bao gồm các thuộc tính:

- idHD (String): ID của hóa đơn chứa món này.
- mon (Mon): biểu diễn món (đã định nghĩa).
- soLuong (int): số lượng món.
- Size (String): kích cỡ món.

Phương thức fromMap() giúp tạo CTHD từ dữ liệu trả về của Supabase, trong đó có cả thông tin món (TraSuaChangTea) được ánh xạ thành đối tượng Mon.

```dart
class CTHD{
  final String idHD;
  final Mon mon;
  final int soLuong;
  final String size;

  CTHD({
    required this.idHD,
    required this.mon,
    required this.size,
    required this.soLuong,
  });

  factory CTHD.fromMap(Map<String, dynamic> map){
    return CTHD(
      idHD: map['idHD'] ?? '',
      mon: Mon.fromJson(map['TraSuaChangTea']),
      soLuong: map['soLuong'] as int,
      size: map['size'],
    );  // CTHD
  }
}
```

Hình 4.21. Class CTHD

4.2.4.2 Invoice Controller

Trong phần quản lý hóa đơn, nhóm đã thiết kế: 2 lớp HoaDonSnapshot và CTHDSnapshot đóng vai trò controller, phụ trách truy xuất và tương tác với cơ sở dữ liệu Supabase, bao gồm các chức năng thêm và lấy dữ liệu.

a. Class HoaDonSnapshot

Lớp HoaDonSnapshot chịu trách nhiệm xử lý hóa đơn tổng, bao gồm việc: truy xuất tất cả hóa đơn của một người dùng (theo idKH) và thêm mới một hóa đơn vào hệ thống qua 2 phương thức getInvoice() và insertInvoice().

- **Phương thức getInvoice():** Truy vấn danh sách các hóa đơn đã được lưu trong Supabase cho người dùng hiện tại, cụ thể:

- Lấy user hiện tại từ Supabase Auth.

- Truy vấn bảng HoaDonChangTea, lọc theo idKH = user.id.

- Map kết quả trả về thành danh sách HoaDon bằng HoaDon.fromMap().

- **Phương thức insertInvoice():** thêm một hóa đơn mới vào Supabase và lấy lại idHD được sinh tự động, cụ thể:

- Gửi dữ liệu hóa đơn thông qua h.toMap(user.id) để tạo Map phù hợp với Supabase.

- Dùng select('idHD') để Supabase trả lại mã idHD vừa thêm vào.

- Trả về idHD dạng String?.

```
class HoaDonSnapshot{
  static Future<List<HoaDon>> getInvoice() async{
    final user = supabase.auth.currentUser;
    var response = await supabase.from('HoaDonChangTea').select("idHD,
     thoiGian, tongTien, ghiChu").eq("idKH", user!.id);
    return response.map((e) => HoaDon.fromMap(e),).toList();
  }

  static Future<String?> insertInvoice(HoaDon h) async{
    final user = supabase.auth.currentUser;
    final response = await supabase.from('HoaDonChangTea').insert(h.toMap
     (user!.id)).select('idHD').single();
    return response['idHD'] as String?;
  }
}
```

Hình 4.22. Class HoaDonSnapshot

b. Class CTHDSnapshot

Lớp CTHDSnapshot quản lý các món trong từng hóa đơn cụ thể (dòng trong bảng CTHDChangTea). Bao gồm truy xuất chi tiết các món trong một hóa đơn, thêm mới dòng chi tiết hóa đơn.

- **Phương thức getDetailInvoice():** truy xuất tất cả các món có trong một hóa đơn, cụ thể:

- Truy vấn bảng CTHDChangTea, kèm liên kết đến bảng TraSuaChangTea (món) và HoaDonChangTea.

- Lọc theo idHD tương ứng.
- Map dữ liệu trả về thành danh sách đối tượng CTHD, với từng món đã có dữ liệu chi tiết.

- **Phương thức insertDetailInvoice():** Thêm một dòng chi tiết hóa đơn vào bảng CTHDChangTea những thông tin liên quan đến món, số lượng, size và ID hóa đơn.

```
class CTHDSnapshot{
  static Future<List<CTHD>> getDetailInvoice(String idHD) async{
    var response = await supabase.from('CTHDChangTea').select("soLuong,
     size, HoaDonChangTea(*), TraSuaChangTea(*)").eq("idHD", idHD);
    return response.map((e) => CTHD.fromMap(e),).toList();
  }

  static Future<void> insertDetailInvoice(CTHD ct) async{
    await supabase.from('CTHDChangTea').insert({
      "idHD": ct.idHD,
      "idMon": ct.mon.id,
      "soLuong": ct.soLuong,
      "size": ct.size
    });
  }
}
```

Hình 4.23. Class CTHDSnapshot

4.2.5 Tìm kiếm (Search)

Lớp lichSuTimKiem được xây dựng dưới dạng GetxController để quản lý logic tìm kiếm. Nó theo dõi dữ liệu nhập vào từ người dùng, lọc danh sách món dựa trên từ khóa, và lưu trữ lịch sử tìm kiếm. Toàn bộ dữ liệu lọc được cập nhật qua dsDeXuatFilter (danh sách món sau khi lọc theo từ khóa), cho phép UI hiển thị kết quả động.

```
class lichSuTimKiem extends GetxController {
  var dstk = <String>[].obs;
  var dsDeXuat = <Mon>[].obs;
  var dsDeXuatFilter = <Mon>[].obs;
  TextEditingController txtSearch = TextEditingController();
```

Hình 4.24. Các biến trong class lichSuTimKiem

Các biến sử dụng trong class:

- dstk (RxList<String>): Lưu lại danh sách lịch sử tìm kiếm từ người dùng → Hiển thị phần "Lịch sử tìm kiếm" khi người dùng chưa nhập gì.
- dsDeXuat (RxList<Mon>): Danh sách toàn bộ món ăn/thức uống được lấy từ cơ sở dữ liệu.
- dsDeXuatFilter (RxList<Mon>): Danh sách món đã được lọc theo từ khóa tìm kiếm.
- txtSearch (TextEditingController): Điều khiển và theo dõi nội dung người dùng nhập vào ô tìm kiếm.

Các phương thức sử dụng:

a. Phương thức onInit()

```
@override
void onInit() {
  super.onInit();
  layMenu();
  txtSearch.addListener(() {
    timKiemTheoTen(txtSearch.text);
  });
}
```

Hình 4.25. Phương thức OnInit()

Phương thức onInit() là một phương thức ghi đè từ GetxController, được gọi tự động khi controller được khởi tạo. Bên trong onInit(), phương thức layMenu() được gọi để lấy toàn bộ danh sách món từ cơ sở dữ liệu thông qua ChangTeaSnapshot.getMon() và lưu trữ vào hai biến dsDeXuat (danh sách gợi ý đầy đủ) và dsDeXuatFilter (danh sách lọc theo từ khóa). Ngoài ra, onInit() cũng gán một listener cho txtSearch, để mỗi khi người dùng nhập dữ liệu vào ô tìm kiếm, phương thức timKiemTheoTen() sẽ được gọi tự động với nội dung tìm kiếm hiện tại.

b. Phương thức layMenu()

```
void layMenu() async {
  List<Mon> allItems = await ChangTeaSnapshot.getMon();
  dsDeXuat.value = allItems;
  dsDeXuatFilter.value = allItems;
}
```

Hình 4.26. Phương thức layMenu()

Phương thức layMenu() được triển khai bất đồng bộ để đảm bảo không chặn giao diện khi tải dữ liệu. Nó truy vấn toàn bộ danh sách món ăn vặt hoặc thức uống, sau đó gán dữ liệu đồng thời cho cả hai danh sách: dsDeXuat để giữ nguyên dữ liệu gốc, và dsDeXuatFilter để phục vụ việc hiển thị dữ liệu đã được lọc.

c. Phương thức addTimKiem()

```
void addTimKiem(String txt) {
  if (!dstk.contains(txt)) {
    dstk.add(txt);
  }
}
```

Hình 4.27. Phương thức addTimKiem()

Phương thức addTimKiem(String txt) có chức năng lưu lại lịch sử tìm kiếm của người dùng. Nếu từ khóa chưa từng được tìm trước đó, phương thức sẽ thêm từ khóa đó vào danh sách dstk, giúp người dùng dễ dàng truy cập lại các từ khóa cũ.

d. Phương thức timKiemTheoTen()

```
void timKiemTheoTen(String txt) {
  final tuKhoa = txt.toLowerCase().trim();
  dsDeXuatFilter.value = dsDeXuat.where(
          (item) => item.ten.toLowerCase().contains(tuKhoa)
  ).toList();
}
```

Hình 4.28. Phương thức timKiemTheoTen()

Phương thức timKiemTheoTen(String txt) là nơi xử lý thuật toán lọc. Từ khóa nhập vào được chuẩn hóa bằng cách chuyển về chữ thường và loại bỏ khoảng trắng

đầu/cuối. Sau đó, danh sách dsDeXuat được lọc bằng điều kiện: tên món (item.ten) có chứa từ khóa tìm kiếm.

Kết quả lọc được gán cho dsDeXuatFilter, từ đó hiển thị lên giao diện ứng với nội dung người dùng đã nhập. Nhờ cơ chế này, hệ thống tìm kiếm trở nên mượt mà và phản hồi nhanh theo từng ký tự mà người dùng nhập.

4.2.6 Cá nhân (Profile)

4.2.6.1 Profile Model

```
class UserChangTea {
  String idKH;
  String tenKH;
  String diaChi;
  String sdt;

  UserChangTea({
    required this.idKH,
    required this.tenKH,
    required this.diaChi,
    required this.sdt,
  });

  Map<String, dynamic> toJson() {
    return {
      'idKH': idKH,
      'tenKH': tenKH,
      'diaChi': diaChi,
      'sdt': sdt,
    };
  }

  factory UserChangTea.fromJson(Map<String, dynamic> json) {
    return UserChangTea(
      idKH: json['idKH'] ?? '',
      tenKH: json['tenKH'] ?? '',
      diaChi: json['diaChi'] ?? '',
      sdt: json['sdt'] ?? '',
    );
  }
}
```

Hình 4.29. Class UserChangTea

Lớp UserChangTea được xây dựng nhằm mục đích biểu diễn thông tin của một khách hàng trong hệ thống, bao gồm các thuộc tính cơ bản như:

- idKH (String): mã khách hàng.
- tenKH (String): tên khách hàng
- diaChi (String): địa chỉ.
- sdt (String): số điện thoại.

Các thuộc tính này đều được khai báo là bắt buộc thông qua từ khóa required trong constructor, đảm bảo rằng mỗi đối tượng được khởi tạo đều đầy đủ thông tin.

Lớp này có hai phương thức hỗ trợ quá trình chuyển đổi dữ liệu: toJson() giúp chuyển đối tượng thành dạng Map để dễ dàng lưu trữ lên cơ sở dữ liệu Supabase hoặc sử dụng với API, còn fromJson() dùng để tạo đối tượng từ dữ liệu JSON lấy từ Supabase. Đặc biệt, việc sử dụng toán tử ?? " trong fromJson() giúp tránh lỗi khi một trường dữ liệu có giá trị null, đảm bảo ứng dụng hoạt động ổn định.

4.2.6.2 Profile Controller

a. Class UserChangTeaSnapshot

```
class UserChangteaSnapshot {
  UserChangTea userChangTea;
  UserChangteaSnapshot({required this.userChangTea});

  static Future<dynamic> updateProfile(UserChangTea newUser) async {
    final supabase = Supabase.instance.client;
    var data = await supabase.from("KhachHangChangTea").update(newUser.toJson()).eq('idKH', newUser.idKH);
    return data;
  }

  static Future<dynamic> inserProfile(UserChangTea newUser) async {
    final supabase = Supabase.instance.client;
    var data = await supabase.from("KhachHangChangTea").insert(newUser.toJson()).eq('idKH', newUser.idKH);
    return data;
  }
}
```

Hình 4.30. Class UserChangTeaSnapshot

Lớp UserChangteaSnapshot đóng vai trò như một lớp trung gian để thực hiện các thao tác liên quan đến việc lưu trữ và cập nhật thông tin khách hàng (UserChangTea) trong cơ sở dữ liệu Supabase. Lớp này chứa hai phương thức tĩnh (static) chính:

- updateProfile(UserChangTea newUser): dùng để cập nhật thông tin của một khách hàng đã có trong bảng KhachHangChangTea. Dữ liệu mới sẽ được chuyển

sang định dạng JSON thông qua phương thức toJson(), sau đó sử dụng hàm .update() kết hợp với điều kiện .eq('idKH', newUser.idKH) để xác định đúng bản ghi cần cập nhật theo mã khách hàng.

- inserProfile(UserChangTea newUser): phương thức này thực hiện thêm mới (insert) một khách hàng vào bảng.

b. Class UserService

```
class UserService {
  final SupabaseClient supabase = Supabase.instance.client;

  Future<UserChangTea?> getCurrentUserChangTea() async {
    final user = supabase.auth.currentUser;
    if (user == null) return null;

    try {
      final data = await supabase
          .from('KhachHangChangTea')
          .select()
          .eq('idKH', user.id)
          .maybeSingle();

      if (data == null) return null;
      return UserChangTea.fromJson(data);
    } catch (e) {
      return null;
    }
  }
}
```

Hình 4.31. Class UserService

Lớp UserService là một lớp dịch vụ với nhiệm vụ chính là truy xuất thông tin người dùng hiện tại từ hệ thống Supabase. Bên trong lớp này, một đối tượng được khởi tạo từ Supabase.instance.client, cho phép tương tác với cơ sở dữ liệu Supabase. Phương thức chính của lớp là getCurrentUserChangTea(), đây là một hàm bất đồng bộ (async) trả về một đối tượng UserChangTea hoặc null tùy thuộc vào việc người dùng hiện tại có đăng nhập và tồn tại trong bảng dữ liệu hay không.

Khi phương thức này được gọi, đầu tiên nó kiểm tra người dùng hiện tại thông qua supabase.auth.currentUser. Nếu không có người dùng nào đang đăng nhập, hàm sẽ trả về null. Ngược lại, nếu có, hệ thống sẽ thực hiện truy vấn tới bảng KhachHangChangTea, tìm dòng dữ liệu mà idKH trùng khớp với user.id từ hệ thống xác thực Supabase. Việc sử dụng .maybeSingle() giúp đảm bảo rằng chỉ một dòng dữ liệu được trả về hoặc null nếu không tìm thấy.

Sau khi truy vấn, nếu không có dữ liệu trả về, hàm tiếp tục trả null. Ngược lại, nếu có dữ liệu, nó sẽ được chuyển thành một đối tượng UserChangTea thông qua phương thức fromJson. Toàn bộ quá trình này được bao bọc trong một khối try-catch để đảm bảo rằng nếu xảy ra lỗi (như mất kết nối mạng hoặc lỗi truy vấn), phương thức sẽ không làm ứng dụng gặp sự cố mà thay vào đó trả về null một cách an toàn.

c. Class ProfileController

Lớp ProfileController trong ứng dụng Flutter có nhiệm vụ điều khiển giao diện hiển thị thông tin hồ sơ người dùng, đóng vai trò như một lớp trung gian giữa dữ liệu người dùng và giao diện người dùng. Lớp này không chứa trạng thái (stateless) và chủ yếu xây dựng hai chế độ hiển thị: giao diện cho khách chưa đăng nhập và giao diện cho người dùng đã đăng nhập.

Phương thức **buildGuestView(BuildContext context)** được sử dụng để hiển thị giao diện dành cho người chưa đăng nhập. Giao diện này gồm biểu tượng tài khoản lớn, thông báo khuyến khích người dùng đăng nhập, cùng với một nút “Đăng nhập” có chức năng chuyển hướng đến trang xác thực PageAuthMilktea.

```
class ProfileController{
  Widget buildGuestView(BuildContext context) {
    return Center(
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 24.0),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Icon(Icons.account_circle, size: 150, color: Colors.amber),
            SizedBox(height: 20),
            Text(
              "Hãy đăng nhập để nhanh chóng đặt món tại quán nhé",
              style: TextStyle(fontSize: 18),
              textAlign: TextAlign.center,
            ),  // Text
            SizedBox(height: 20),
            SizedBox(
              width: 150,
              child: ElevatedButton.icon(
                icon: Icon(Icons.login),
                label: Text("Đăng nhập"),
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(builder: (_) => PageAuthMilktea()),
                  );
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.amberAccent,
                  foregroundColor: Colors.black,
                  padding: EdgeInsets.symmetric(vertical: 14),
                  textStyle: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
```

Hình 4.32. Phương thức buildGuestView()

Ngược lại, phương thức **buildProfileView(BuildContext context, User user, UserChangTea userChangTea)** được dùng khi người dùng đã đăng nhập. Nó hiển thị thông tin chi tiết như email, tên khách hàng, địa chỉ, và số điện thoại, được trình bày dưới dạng danh sách có phân tách rõ ràng bằng các dòng Divider. Ngoài ra, giao diện này cũng tích hợp hình đại diện khách hàng dạng tròn với đường viền vàng. Giao diện cung cấp hai chức năng nổi bật: truy cập lịch sử mua hàng thông qua PurchaseHistory và cập nhật thông tin thông qua PageUpdateProfile. Đặc biệt, còn có nút "Đăng xuất", khi được nhấn sẽ thực hiện thao tác đăng xuất khỏi Supabase, đồng thời làm mới lại thông tin giỏ hàng thông qua CartController.

Cuối cùng, phương thức phụ trợ inforRow(String label, String value) giúp tái sử dụng đoạn mã hiển thị thông tin theo cặp nhãn–giá trị, đảm bảo giao diện thống nhất và dễ bảo trì.

```
Widget buildProfileView(BuildContext context, User user, UserChangTea userChangTea) {
  return SingleChildScrollView(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        SizedBox(height: 10),
        Center(
          child: Container(
            width: 150,
            height: 150,
            decoration: BoxDecoration(
              shape: BoxShape.circle,
              border: Border.all(color: Colors.amber, width: 4),
              image: DecorationImage(
                image: AssetImage("asset/Images/avatar.jpg"),
                fit: BoxFit.cover,
              ),  // DecorationImage
            ),  // BoxDecoration
          ),  // Container
        ),  // Center
        SizedBox(height: 20),
        Divider(),
        Text("Thông tin khách hàng", style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
        Divider(),
        inforRow("Email: ", user.email ?? ""),
        inforRow("Khách hàng: ", userChangTea.tenKH),
        inforRow("Địa chỉ: ", userChangTea.diaChi),
        inforRow("Số điện thoại: ", userChangTea.sdt),
        Divider(),
        Row(
```

Hình 4.33. Minh hoạ phương thức buildProfileView()

4.3 THIẾT KẾ GIAO DIỆN

4.3.1 Giao diện trang chủ của ứng dụng

Hình 4.34. Giao diện trang chủ

- Giao diện *Trang chủ* của Ứng dụng đặt hàng trà sữa ChangTea bao gồm:

1. Thanh tiêu đề (AppBar): tên thương hiệu "Trà sữa ChangTea", biểu tượng Giỏ hàng nằm ở góc phải để người dùng truy cập nhanh vào giỏ hàng.

2. Phần thân:

- Trên cùng là ảnh động giới thiệu các món tiêu biểu của quán.

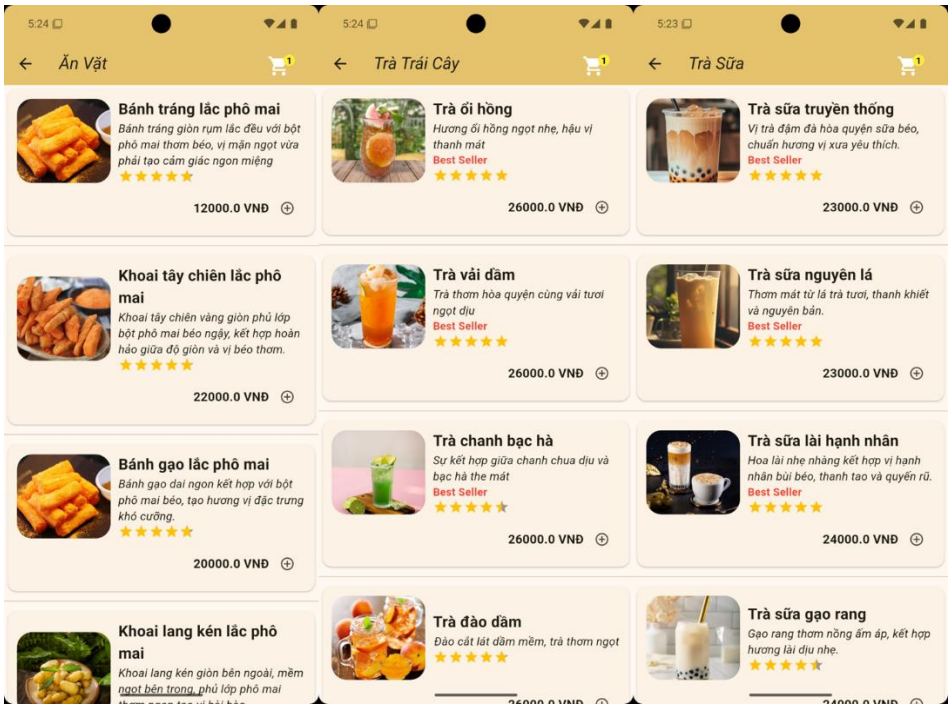
- Bên dưới chia thành các tab nhóm sản phẩm: Trà Sữa, Trà Trái Cây, Ăn Vặt; hiển thị một số món best seller của ChangTea ở mỗi loại sản phẩm theo dạng danh sách.

+ Người dùng có thể bấm vào "*Xem thêm*" bên phải ở tab mỗi loại để xem hết tất cả các món của loại sản phẩm đó.

+ Ngoài ra còn có thể xem hình ảnh minh họa của món, mô tả ngắn về món (tên món ăn, thông tin món, giá tiền,…).

3. Thanh điều hướng bên dưới: gồm 3 mục Trang chủ, Tìm kiếm, Cá nhân. Khi bấm vào, sẽ di chuyển đến giao diện mà người dùng vừa thao tác.

4.3.2 Giao diện trang hiển thị danh sách các món của quán

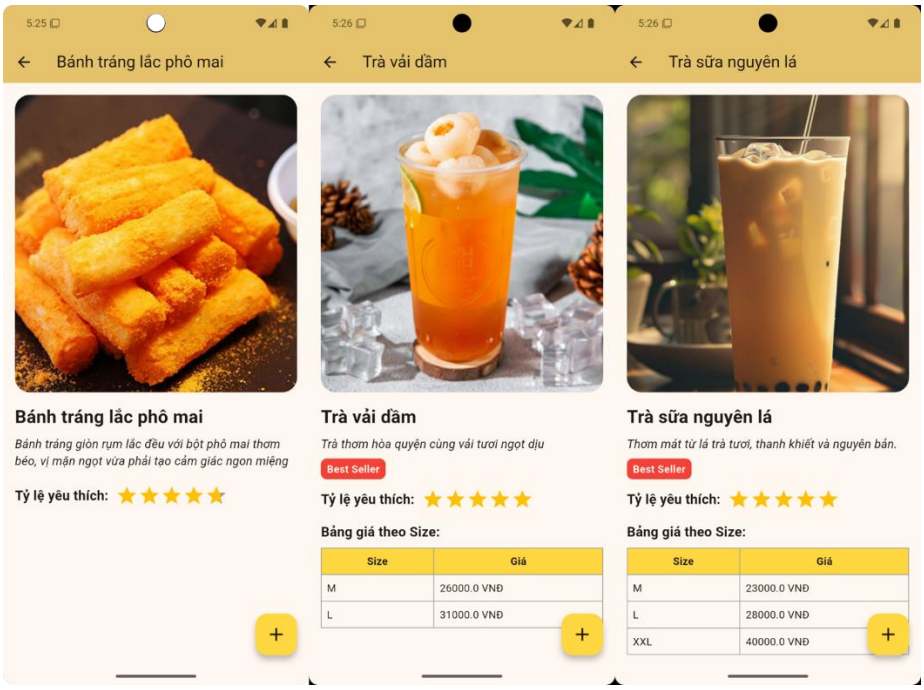


Hình 4.35. Giao diện hiển thị các món của quán

Giao diện *Hiển thị các món* của ChangTea theo từng tab: Trà Sữa, Trà Trái Cây, Ăn Vặt, tên các tab này hiển thị trên AppBar. Mỗi món gồm hình ảnh minh họa, thông tin về món (mô tả món, đánh giá, giá tiền của món).

Người dùng có thể thêm vào giỏ hàng bằng cách bấm vào dấu "+" bên phải giá tiền mỗi món, hoặc bấm vào món để xem thông tin chi tiết món đã chọn.

4.3.3 Giao diện trang chi tiết món



Hình 4.36. Giao diện chi tiết món

Khi muốn xem thông tin chi tiết của món, chỉ cần bấm vào món đó:

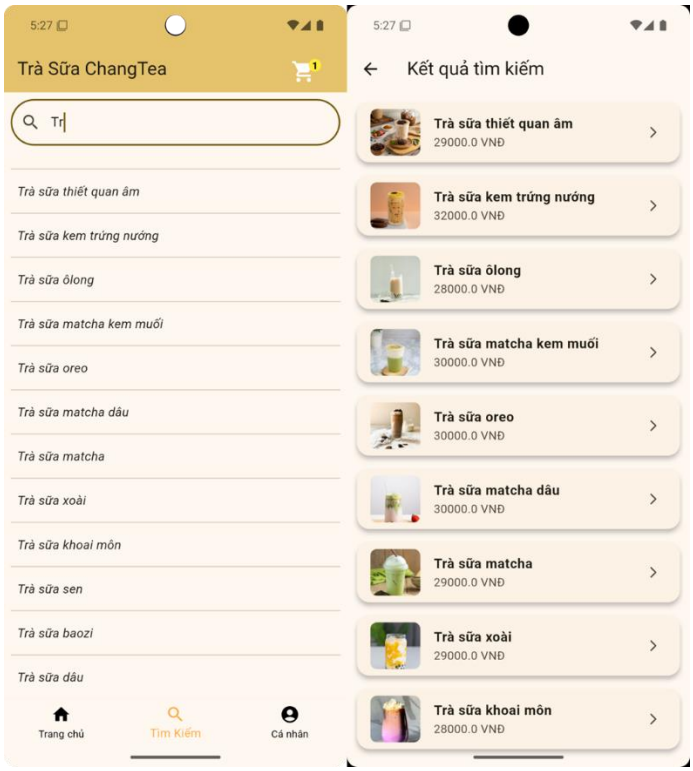
- Tên món đã chọn để xem sẽ hiển thị trên AppBar.

- Người dùng sẽ thấy được hình ảnh món rõ ràng, sắc nét hơn.

- Các thông tin của món được hiển thị đầy đủ: tên món được in đậm, mô tả chi tiết món, đánh giá của khách hàng (tỷ lệ yêu thích) và có nhãn best seller nếu là món bán chạy, tất cả size đang có của món (trừ đồ ăn vặt).

- Dưới cùng bên phải là nút thêm món vào giỏ hàng (+).

4.3.4 Giao diện trang tìm kiếm



Hình 4.37. Giao diện trang Tìm kiếm

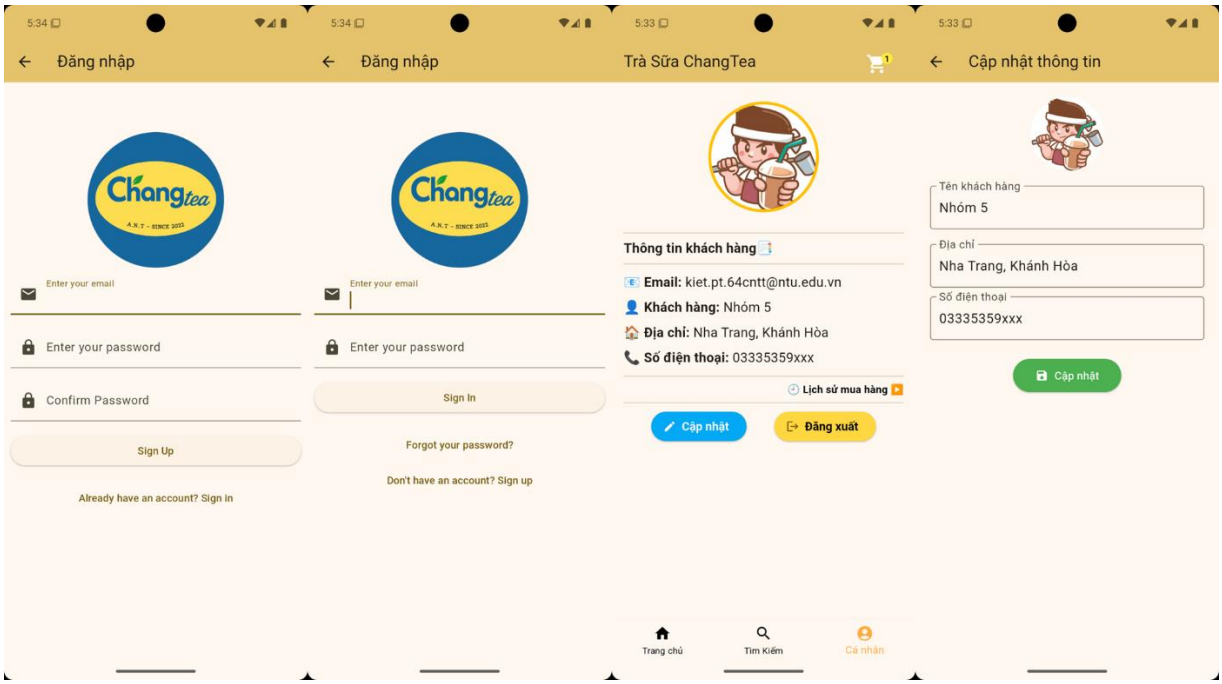
Giao diện trang *Tìm kiếm* giúp người dùng tìm kiến nhanh món mình đang cần, tiết kiệm thời gian và thao tác nhanh hơn, bao gồm:

- Thanh tìm kiếm nằm ở trên cùng, có biểu tượng kính lúp và placeholder “Tìm kiếm”.

- Khi người dùng nhập, danh sách các món (dạng văn bản) phù hợp sẽ hiện ra theo thời gian thực.

- Sau khi bấm tìm kiếm, *Kết quả tìm kiếm* sẽ hiện ra, mỗi item gồm ảnh món, tên và giá. Để xem chi tiết, người dùng bấm vào biểu tượng “>”.

4.3.5 Giao diện trang đăng nhập, đăng ký và thông tin khách hàng



Hình 4.38. Giao diện Đăng nhập, Đăng ký tài khoản khách hàng

- Giao diện *Đăng ký* và *Đăng nhập* giúp người dùng tạo tài khoản mới (nếu chưa có) hoặc đăng nhập (nếu đã có tài khoản), bao gồm:

+ Logo quán trà sữa ChangTea ở chính giữa.

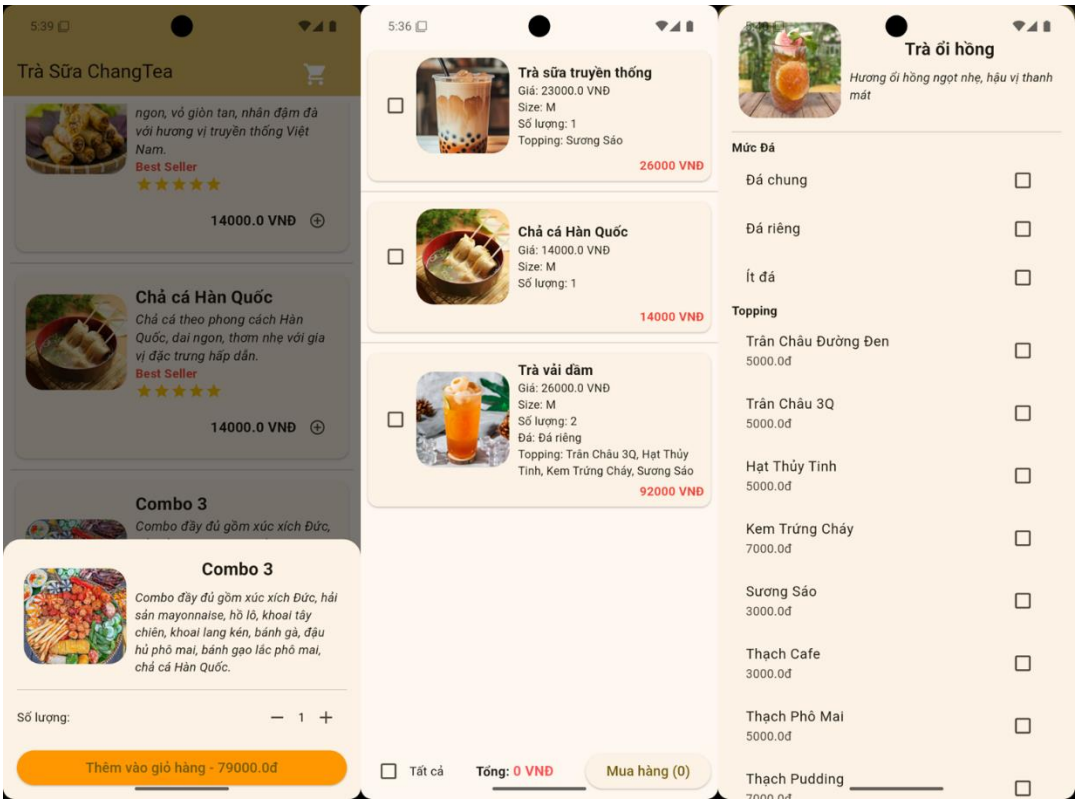
+ Các trường nhập liệu gồm: Email, password và xác nhận mật khẩu (trong trường hợp đăng ký tài khoản mới).

+ Có điều hướng rõ ràng như chuyển sang trang *Đăng ký* (Sign in) nếu chưa có tài khoản, hoặc sang trang khôi phục mật khẩu nếu đã quên mật khẩu trước đó.

- Sau khi đăng nhập thành công, giao diện *Thông tin khách hàng* sẽ hiện ra, hiển thị đầy đủ thông tin, bao gồm: tên email, tên khách hàng, địa chỉ và số điện thoại. Người dùng có thể cập nhật thông tin của mình hoặc đăng xuất nếu muốn. Ngoài ra còn có thể xem các đơn hàng mình đã mua trước đó bằng *Lịch sử mua hàng*.

- Khi bấm vào *Cập nhật bên dưới* thông tin khách hàng, người dùng có thể cập nhật các thông tin của mình như: tên khách hàng, địa chỉ, số điện thoại.

4.3.6 Giao diện trang giỏ hàng



Hình 4.39. Giao diện trang Giỏ hàng

- Giỏ hàng là nơi lưu trữ các món mà người dùng đã thêm, thuận tiện cho việc mua hàng, xem lại khi có nhu cầu mà không cần phải tìm kiếm lại từ đầu.

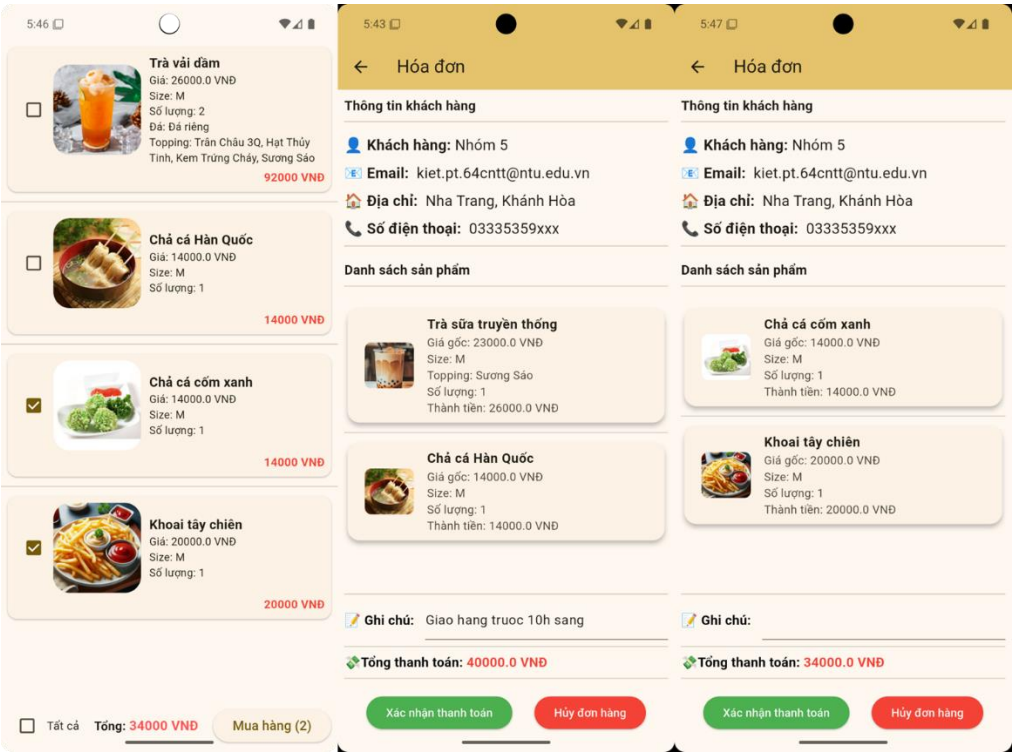
- Sau khi xem các món, người dùng muốn thêm món vào giỏ hàng sẽ bấm vào dấu "+" bên cạnh giá tiền của món đó. Một bảng chọn sẽ trượt lên từ bên dưới, cho phép người dùng chọn topping, mức đá (nếu là trà trái cây hoặc trà sữa), và size theo ý muốn.

- Khi xác nhận thêm vào giỏ hàng, biểu tượng giỏ hàng trên AppBar sẽ tăng số lượng, và khi bấm vào đó, danh sách các món mà người dùng thêm vào sẽ xuất hiện tại đây, gồm các thông tin như:

  + Mỗi món có tên món, size, số lượng, giá…, một checkbox phía trước đánh dấu có lựa chọn món đó để mua hay không.

  + Bên dưới là checkbox *Tất cả,* khi tick vào đây, tất cả các món trong giỏ hàng đều được lựa chọn hết một cách nhanh chóng mà không phải tick từng cái một, theo đó *Tổng* hiển thị số tiền bạn cần phải trả nếu mua hàng. Bên cạnh nút *Mua hàng* còn có số lượng món người dùng đã chọn. Khi bấm *Mua hàng*, giao diện *Hóa đơn* sẽ xuất hiện.

4.3.7 Giao diện thanh toán hóa đơn



Hình 4.40 Giao diện Hóa đơn

Khi bấm mua hàng từ *Giỏ hàng,* giao diện *Hóa đơn* sẽ xuất hiện giúp người dùng kiểm tra lại một lần nữa thông tin cá nhân của mình để đặt hàng, các mặt hàng mình muốn mua có đúng hay chưa, giao diện bao gồm:

+ Thanh tiêu đề "Hóa đơn".

+ Thông tin khách hàng gồm tên khách hàng, email, địa chỉ, số điện thoại.

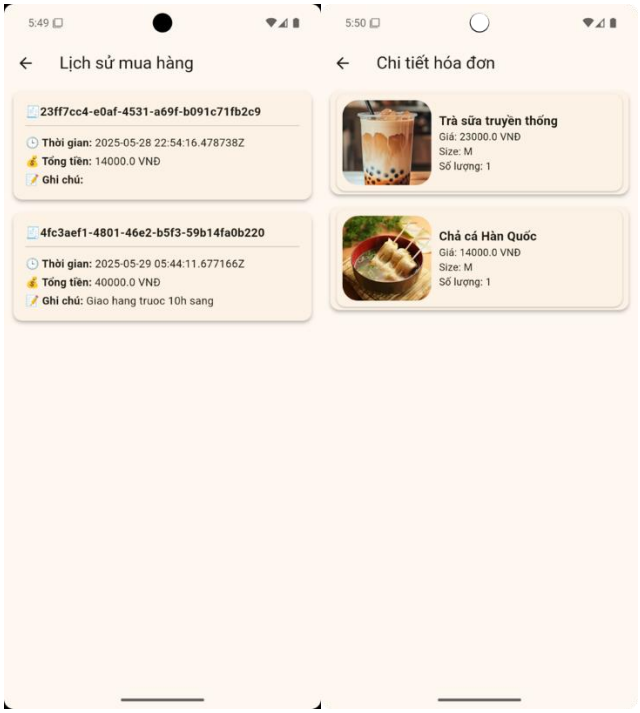
+ Danh sách các sản phẩm muốn mua.

+ Ghi chú mà khách hàng muốn gửi đến cửa hàng (vd: cho ít ngọt, không bỏ thạch,…)

+ Tổng số tiền cần phải thanh toán cho đơn hàng.

+ 2 nút *Xác nhận thanh toán* nếu khách hàng đã chắc chắn mua và *Hủy đơn hàng* nếu không muốn mua nữa.

4.3.8 Giao diện lưu lịch sử hóa đơn, chi tiết hóa đơn



Hình 4.41. Giao diện Lịch sử mua hàng, Chi tiết hóa đơn

Khi người dùng muốn xem lại những đơn hàng mình đã mua, họ có thể vào trang *Cá nhân* và bấm vào *Lịch sử mua hàng*, lúc này tất cả các đơn hàng đã mua sẽ hiện ở đây.

Giao diện bao gồm nhiều đơn hàng, mỗi đơn hàng gồm những thông tin mã đơn hàng (hóa đơn), thời gian cụ thể đặt hàng, tổng số tiền của đơn hàng và ghi chú (nếu có).

Nếu muốn biết cụ thể đã mua những món gì trong từng đơn hàng đã mua, người dùng chỉ cần bấm vào đơn hàng đó, các món cụ thể sẽ hiện ra trong *Chi tiết hóa đơn* gồm: hình ảnh món, giá tiền, size và số lượng.

Chương 5. KẾT LUẬN

5.1 ƯU ĐIỂM - KẾT QUẢ ĐẠT ĐƯỢC

Trong quá trình thực hiện bài tập lớn "Ứng dụng đặt trà sữa ChangTea", nhóm em đã đạt được hầu như các yêu cầu cơ bản mà ban đầu nhóm đã đề ra, cụ thể:

- Ứng dụng có giao diện dễ nhìn, dễ hiểu, màu sắc hài hòa, phù hợp với phong cách của quán. Ứng dụng thể hiện các nội dung một cách trực quan, thân thiện, dễ dàng sử dụng đối với người dùng.

- Ứng dụng đã hoàn thiện tính năng hiển thị danh sách các món trà sữa, trà trái cây và đồ ăn vặt, kèm theo đầy đủ thông tin như tên, hình ảnh, mô tả và giá bán. Người dùng có thể dễ dàng nhấn vào từng món để xem chi tiết, giúp trải nghiệm lựa chọn trở nên trực quan và sinh động hơn.

- Ứng dụng hỗ trợ người dùng tìm kiếm các món trà sữa một cách nhanh chóng và thuận tiện. Đồng thời, hệ thống cũng hiển thị các gợi ý liên quan đến từ khóa tìm kiếm, giúp người dùng dễ dàng khám phá thêm những món phù hợp với sở thích.

- Ứng dụng hỗ trợ lưu giỏ hàng riêng biệt cho từng tài khoản đăng nhập, giúp người dùng dễ dàng quản lý các món đã chọn. Người dùng có thể linh hoạt thêm món yêu thích vào giỏ hoặc xóa những món không còn nhu cầu.

- Ứng dụng cho phép hiển thị hóa đơn mua hàng một cách rõ ràng. Khi hoàn tất đơn hàng, ứng dụng sẽ lưu và hiển thị đầy đủ hóa đơn với các thông tin chi tiết (mã số, thời gian và số lượng) và chi tiết hóa đơn của từng đơn hàng.

5.2 NHƯỢC ĐIỂM - HẠN CHẾ

Bên cạnh những mục tiêu đã đặt được, "Ứng dụng đặt trà sữa ChangTea" của nhóm em vẫn còn một cố hạn chế tồn động chưa thể hoàn thành, cụ thể:

- Hiện tại, ứng dụng vẫn chưa hỗ trợ tính năng cho phép người dùng đánh giá các món hoặc phản hồi sau khi đơn hàng được giao thành công, điều này làm giảm khả năng tương tác và thu thập phản hồi từ khách hàng.

- Chức năng tìm kiếm còn khá đơn giản, chưa được tối ưu để mang lại kết quả đa dạng và sát với nhu cầu người dùng.

- Giỏ hàng chưa hỗ trợ chỉnh sửa trực tiếp các thông tin như số lượng, kích cỡ (size), hay các yêu cầu đặc biệt đi kèm món, gây bất tiện trong quá trình điều chỉnh đơn hàng trước khi thanh toán.

5.3 ĐỀ XUẤT PHÁT TRIỂN

Qua quá trình tìm hiểu và phát triển "Ứng dụng đặt trà sữa ChangTea", nhóm em có một số đề xuất để có thể phát triển ứng dụng một cách đa dạng và mới mẽ hơn.

- Ứng dụng cho phép người dùng nhắn tin và trao đổi trực tiếp với quán, góp phần tăng cường sự tương tác giữa khách hàng và nhân viên. Tính năng này hỗ trợ tư vấn món phù hợp theo nhu cầu, đồng thời giải đáp thắc mắc kịp thời cho khách hàng.

- Người dùng có thể đánh giá món ăn và gửi phản hồi sau khi đơn hàng được giao thành công, giúp quán thu thập ý kiến và nâng cao chất lượng dịch vụ. Bên cạnh đó, ứng dụng còn có thể tích điểm cho mỗi đơn hàng để đổi lấy Voucher hoặc thăng hạng thành viên thân thiết.

- Ứng dụng hỗ trợ đa dạng phương thức thanh toán, không chỉ giới hạn ở tiền mặt mà còn tích hợp các hình thức điện tử như ví Momo, ZaloPay,... nhằm mang lại sự tiện lợi tối đa cho khách hàng.

- Khách hàng có thể theo dõi trạng thái đơn hàng theo thời gian thực, từ lúc đặt món cho đến khi giao hàng, giúp kiểm soát tiến trình đơn hàng một cách rõ ràng và an tâm hơn khi sử dụng dịch vụ.

- Ngoài hình thức đăng nhập bằng Email, ứng dụng còn hỗ trợ tạo tài khoản và đăng nhập thông qua số điện thoại, Google hoặc Facebook. Điều này giúp tiếp cận dễ dàng hơn với nhóm khách hàng không sử dụng email, đồng thời rút ngắn quy trình đăng ký, mang lại trải nghiệm thuận tiện và linh hoạt hơn cho người dùng.

5.4 KẾT LUẬN

Qua quá trình thực hiện bài tập lớn xây dựng "Ứng dụng đặt trà sữa ChangTea", nhóm em đã vận dụng được các kiến thức trên lớp, hiểu và áp dụng được ngôn ngữ Dart, Flutter cùng với nền tảng Android Studio để thực hiện bài tập. Nhóm còn vận dụng một số kiến thức của các môn học liên quan khác như cơ sở dữ liệu để lưu trữ và truy vấn dữ liệu một cách hiệu quả, phân tích được hệ thống và đề ra được hướng đi phù hợp nhất.

Bên cạnh đó, nhóm cũng tham khảo một số bài tập thực hành trên lớp và các trang web trên mạng để tiếp thu và phát triển nên thành phẩm cuối cùng của nhóm.

Trong quá trình thực hiện bài tập lớn cuối kỳ, nhóm em đã nhận được sự hỗ trợ tận tình và góp ý từ thầy Huỳnh Tuấn Anh. Tuy nhiên, do còn hạn chế về kinh nghiệm thực tế nên bài báo cáo của nhóm có thể vẫn còn một số sai sót. Nhóm em rất mong nhận được thêm ý kiến đóng góp từ thầy để có thể tiếp thu, rút kinh nghiệm và cải thiện cho những dự án bài tập lớn ở các môn học khác trong thời gian tới.

Nhóm em xin chân thành cảm ơn thầy!

TÀI LIỆU THAM KHẢO

[1] https://vbee.vn/blog/google/android-studio/?utm_source=BL&utm_medium=CPC&utm_term, (vbee.vn, 02/03/2025, *Android Studio là gì? Cách cài đặt và thiết lập trên Windows, macOS và Linux*), truy cập ngày 17/05/2025.

[2] https://aws.amazon.com/what-is/flutter/#:~:text=Flutter%20is%20an%20open, (aws.amazon.com, *Flutter là gì?*), truy cập ngày 16/05/2025.

[3] https://topdev.vn/blog/flutter-la-gi/ (topdev.vn, 19/08/2018, *Flutter là gì? Ưu điểm vượt trội và cơ hội việc làm hấp dẫn*), truy cập ngày 16/05/2025.

[4] https://blog.flutter.wtf/top-mobile-apps-built-with-flutter-framework/ (blog.flutter.wtf, 28/03/2024, *Top Famous Apps Built with Flutter Framework*), truy cập ngày 17/05/2025.

[5] https://tokyotechlab.com/blogs/ngon-ngu-dart-la-gi (tokyotechlab.com, 23/07/2024, *What is Dart Programming language? Features and applications of Dart*), truy cập ngày 17/05/2025.

[6] https://bigdataclouds.org/supabase-overview/ (bigdataclouds.org, 07/01/2025, *Supabase Overview*), truy cập ngày 17/05/2025.

[7] https://www.postgresql.org/about/ (postgresql.org, *What is PostgreSQL?*), truy cập ngày 17/05/2025.

[8] https://dev.to/jehnz/what-is-supabase-how-to-integrate-it-with-your-react-application-5hea (dev.to, 04/06/2024, *What is Supabase? How to Integrate It with Your React Application*), truy cập ngày 17/05/2025.

[9] https://plugins.jetbrains.com/plugin/6351-dart# (plugins.jetbrains.com, *Dart Plugin*), truy cập ngày 17/05/2025.

[10] https://plugins.jetbrains.com/plugin/9212-flutter (plugins.jetbrains.com, *Flutter Plugin*), truy cập ngày 17/05/2025.

[11] https://plugins.jetbrains.com/plugin/13666-flutter-intl (plugins.jetbrains.com, *Flutter Intl Plugin*), truy cập ngày 17/05/2025.

[12] https://plugins.jetbrains.com/plugin/12429-dart-data-class (plugins.jetbrains.com, *Dart Data Class Plugin*), truy cập ngày 17/05/2025.

[13] https://plugins.jetbrains.com/plugin/12127-dart-json-serialization-generator (plugins.jetbrains.com, *Dart Json Serialization Generator Plugin*), truy cập ngày 17/05/2025.

[14] https://raman04.hashnode.dev/using-setstate-for-simple-state-management-in-flutter (raman04, 20/09/3/2013, *Using setState for Simple State Management in Flutter*), truy cập ngày 17/05/2025.

[15] https://www.geeksforgeeks.org/flutter-getx-state-management-library/ (geeksforgeeks, 25/04/2025, *Flutter - GetX State Management Library*), truy cập ngày 17/05/2025.

[16] https://pub.dev/packages/get (pub.dev, 13/12/2025, *get 4.7.2*), truy cập ngày 17/05/2025.