**TRƯỜNG ĐẠI HỌC THUỶ LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN: PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG**

ĐỀ TÀI: ỨNG DỤNG TLU CANTEEN CONNECT

(KẾT NỐI NHÀ ĂN TLU)

***Giảng viên hướng dẫn: Ths. Kiều Tuấn Dũng***

***Nhóm thực hiện: Nhóm 09 – 64 CNTT3***

|  |  |
| --- | --- |
| **Mã sinh viên** | **Họ tên sinh viên** |
| 2251061913 | Nguyễn Tuấn Tú |
| 2251061904 | Lưu Văn Trung |
| 2251061809 | Vũ Hồng Khoa |
| 2251061912 | Nguyễn Đức Tú |

**Hà Nội, tháng 6 năm 2025**

**LỜI MỞ ĐẦU**

Hiện nay, sinh viên và cán bộ trong trường thường gặp khó khăn khi đến nhà ăn vào giờ cao điểm, dẫn đến tình trạng phải chờ đợi lâu hoặc hết món. Đồng thời, việc nhà ăn chuẩn bị nguyên liệu cũng gặp khó khăn khi không biết nhu cầu thực tế. Từ đó, nhóm chúng em đã nảy sinh ra ý tưởng phát triển một ứng dụng có tên là “TLU Canteen Connect - giúp kết nối sinh viên, cán bộ với nhà ăn trường một cách tiện lợi và hiện đại hơn”.

TLU Canteen Connet (Kết nối nhà ăn TLU) là một hệ thống trực tuyến cung cấp thông tin về thực đơn hàng ngày/hàng tuần và cho phép người dùng đặt món trước – giúp nhà ăn chuẩn bị nguyên liệu hợp lý, hạn chế lãng phí, đồng thời giúp sinh viên tiết kiệm thời gian và tránh chen lấn.

Rất mong nhận được những ý kiến đóng góp từ thầy và các bạn để giúp dự án của chúng em ngày càng hoàn thiện hơn.

Xin chân thành cảm ơn!

Mục lục

[**Chương 1: Tổng quan về đề tài** 4](#_Toc201678698)

[**1.1.** **Giới thiệu về đề tài** 4](#_Toc201678699)

[**1.2.** **Mục tiêu của đề tài** 4](#_Toc201678700)

[*1.2.1.* *Về phía sinh viên* 4](#_Toc201678701)

[*1.2.2.Về phía quản lý nhà ăn* 4](#_Toc201678702)

[**1.3.** **Phạm vi đề tài** 5](#_Toc201678703)

[1.3.1.Đối với người dùng là sinh viên 5](#_Toc201678704)

[1.3.2. Đối với người quản trị 6](#_Toc201678705)

[**1.4.** **Phân chia nhiệm vụ** 7](#_Toc201678706)

[**Chương 2: Kiến trúc và công nghệ** 8](#_Toc201678707)

[**2.1. Kiến trúc hệ thống** 8](#_Toc201678708)

[**2.2.Giới thiệu về Công nghệ phát triển** 9](#_Toc201678709)

[**Chương 3: Xây dựng ứng dụng** 10](#_Toc201678710)

[**3.1. Thiết kế Figma** 10](#_Toc201678711)

[**3.2.Thiết kế cơ sở dữ liệu** 12](#_Toc201678712)

[**3.3.Giao diện người dùng** 15](#_Toc201678713)

[**3.4.Giao diện quản trị viên** 25](#_Toc201678714)

# **Chương 1: Tổng quan về đề tài**

* 1. **Giới thiệu về đề tài**

Trong bối cảnh công nghệ số đang phát triển mạnh mẽ, việc ứng dụng các giải pháp thông minh vào đời sống sinh viên ngày càng trở nên cần thiết, đặc biệt là trong các hoạt động phục vụ thiết yếu như ăn uống tại canteen.

Dự án **TLU Canteen Connect** ra đời với mục tiêu cách mạng hóa trải nghiệm ăn uống của sinh viên tại canteen Trường Đại học Thủy Lợi. Hiện tại, sinh viên thường xuyên phải đối mặt với các vấn đề như xếp hàng dài, khó khăn trong việc tìm kiếm thông tin về thực đơn, hoặc thanh toán bất tiện.

**TLU Canteen Connect** là một hệ thống toàn diện, tích hợp các tính năng đặt món trực tuyến, thanh toán không tiền mặt, hiển thị thực đơn chi tiết, giúp tối ưu hóa quy trình phục vụ và nâng cao sự hài lòng cho cả sinh viên lẫn ban quản lý canteen. Với **TLU Canteen Connect**, chúng tôi hướng tới việc xây dựng một môi trường canteen hiện đại, tiện lợi và hiệu quả, góp phần cải thiện chất lượng cuộc sống sinh viên và tạo dựng một cộng đồng học đường năng động hơn.

* 1. **Mục tiêu của đề tài**
     1. *Về phía sinh viên*
* Tối ưu hóa thời gian đặt và nhận món
* Cung cấp thông tin thực đơn minh bạch và đầy đủ
* Nâng cao sự tiện lợi trong thanh toán
* Cải thiện chất lượng dịch vụ

### *1.2.2.Về phía quản lý nhà ăn*

* Nâng cao hiệu quả quản lý đơn hàng
* Tối ưu hóa quy trình bán hàng và chuẩn bị
* Cải thiện chất lượng dịch vụ dựa trên dữ liệu
  1. **Phạm vi đề tài**

### 1.3.1.Đối với người dùng là sinh viên

*1.3.1.1.Đăng nhập/Đăng ký*:

* Tạo tài khoản mới bằng email, số điện thoại hoặc tài khoản sinh viên.
* Đăng nhập vào ứng dụng.
* Quên mật khẩu/Đặt lại mật khẩu.

*1.3.1.2.Xem thực đơn*

* Hiện thị thực đơn theo ngày
* Xem chi tiết món ăn(tên, giá, mô tả, thành phần, hình ảnh)
* Hiện thị các món ăn đặc biệt, món mới, bán chạy

*1.3.1.3.Đặt món*

* Thêm món vào giỏ hàng
* Điều chỉnh số lượng
* Xem tổng tiền

*1.3.1.4.Thanh toán*

* Thanh toán trực tiếp
* Thanh toán trực tuyến

*1.3.1.5.Thông báo*

* Nhận thông báo về đơn hàng
* Nhận thông báo về hệ thống (đóng/mở, bảo trì, …)

*1.3.1.6. Người dùng*

* Xem thông tin cá nhân
* Lịch sử đặt món

### 1.3.2. Đối với người quản trị

*1.3.2.1. Quản lý thực đơn*

* Thêm, xóa , sửa món ăn
* Cập nhật giá cả, mô tảm, hình ảnh món ăn
* Quản lý danh mục món ăn
* Đặt trạng thái hết/còn hàng cho món ăn

*1.3.2.2.Quản lý đơn hàng*

* Xem danh sách đơn hàng mới, đang xử lý, đã hoàn thành
* Cập nhật trạng thái đơn hàng (xác nhận, đang chuẩn bị, hoàn thành)
* Xem chi tiết từng đơn hàng(món ăn, số lượng, thông tin khách hàng, thanh toán)
  1. **Phân chia nhiệm vụ**

|  |  |
| --- | --- |
| **Thành viên** | **Nhiệm vụ** |
| **Nguyễn Tuấn Tú** | ***Giao diện của quản trị viên***  + Thiết kế Logo  +Chức năng xem thực đơn mà không cần đăng nhập  + Chức năng quản lý món ăn  + Chức năng quản lý đơn hàng đặt trước |
| **Lưu Văn Trung** | ***Giao diện của quản trị viên***  + Chức năng quản lý thực đơn  + Đăng nhập/ Đăng ký  + Quên mật khẩu  + Kết nối Firebase |
| **Vũ Hồng Khoa** | ***Giao diện người dùng***  +Giao diện trang chủ khi chưa đăng nhập/ đăng ký  +Đăng nhập/Đăng ký  +Chi tiết món ăn  +Đặt món  +Giỏ hàng  +Thanh toán |
| **Nguyễn Đức Tú** | ***Giao diện người dùng***  +Giao diện trang chủ khi đã đăng nhập/đăng ký  +Thực đơn trong ngày (Đồ ăn/ đồ uống)  + Thông báo  + Người dùng (Thông tin cá nhân, Lịch sử đặt món) |

# **Chương 2: Kiến trúc và công nghệ**

## **2.1. Kiến trúc hệ thống**

Ứng dụng được xây dựng theo mô hình client-server với sự hỗ trợ của dịch vụ Firebase. Kiến trúc tổng thể của hệ thống gồm các thành phần chính sau:

*2.1.1.Ứng dụng Client (Android App)*

* Là nơi người dùng tương tác trực tiếp để thực hiện các chức năng
* Giao diện người dùng (UI) được phát triển với ngôn ngữ Java, tuân thủ nguyên tắc Material Design, tối ưu trải nghiệm trên thiết bị di động
* Ứng dụng sử dụng Firebase để kết nối với các dịch vụ backend

*2.1.2.FireBase Authentication (Xác thực người dùng)*

* Cung cấp cơ chế xác thực an toàn, hỗ trợ đăng ký/đăng nhập bằng email, Google.
* Đảm bảo rằng mỗi người dùng có không gian lưu trữ dữ liệu riêng biệt, bảo mật.

*2.1.3.FireBase Cloud Firestore (Cơ sở dữ liệu)*

* Là cơ sở dữ liệu Realtime NoSQL, lưu trữ thông tin người dùng, giao dịch, ngân sách, nhắc nhở dưới dạng collection – document.
* Hỗ trợ subcollection, cho phép tổ chức dữ liệu linh hoạt, phù hợp với việc lưu trữ các giao dịch hoặc ngân sách riêng biệt cho từng người dùng.
* Cung cấp khả năng đồng bộ hóa thời gian thực (Realtime Sync), hỗ trợ trải nghiệm liền mạch khi sử dụng đa thiết bị.

## **2.2.Giới thiệu về Công nghệ phát triển**

*2.2.1 Firebase*

* Firebase Authentication: Cung cấp hệ thống xác thực người dùng mạnh mẽ, hỗ trợ đa nền tảng như Android, iOS, Web. Firebase Authentication dễ dàng tích hợp với Android SDK, hỗ trợ đăng nhập bằng Email/Password, Google, Facebook, v.v.
* Firebase Firestore: Là cơ sở dữ liệu NoSQL thời gian thực, hỗ trợ lưu trữ dữ liệu dưới dạng collection/document. Firestore dễ dàng mở rộng, hỗ trợ truy vấn linh hoạt và đảm bảo an toàn nhờ vào hệ thống bảo mật Security Rules. Ngoài ra, Firestore còn hỗ trợ đồng bộ hóa dữ liệu theo thời gian thực giữa các thiết bị.

*2.2.2.Android SDK*

* Ngôn ngữ lập trình: Java.
* Bộ công cụ phát triển chính: Android Studio – môi trường phát triển chính thức do Google cung cấp, hỗ trợ lập trình giao diện (UI), quản lý vòng đời ứng dụng (Lifecycle), và tích hợp trực tiếp với Firebase qua Plugin.

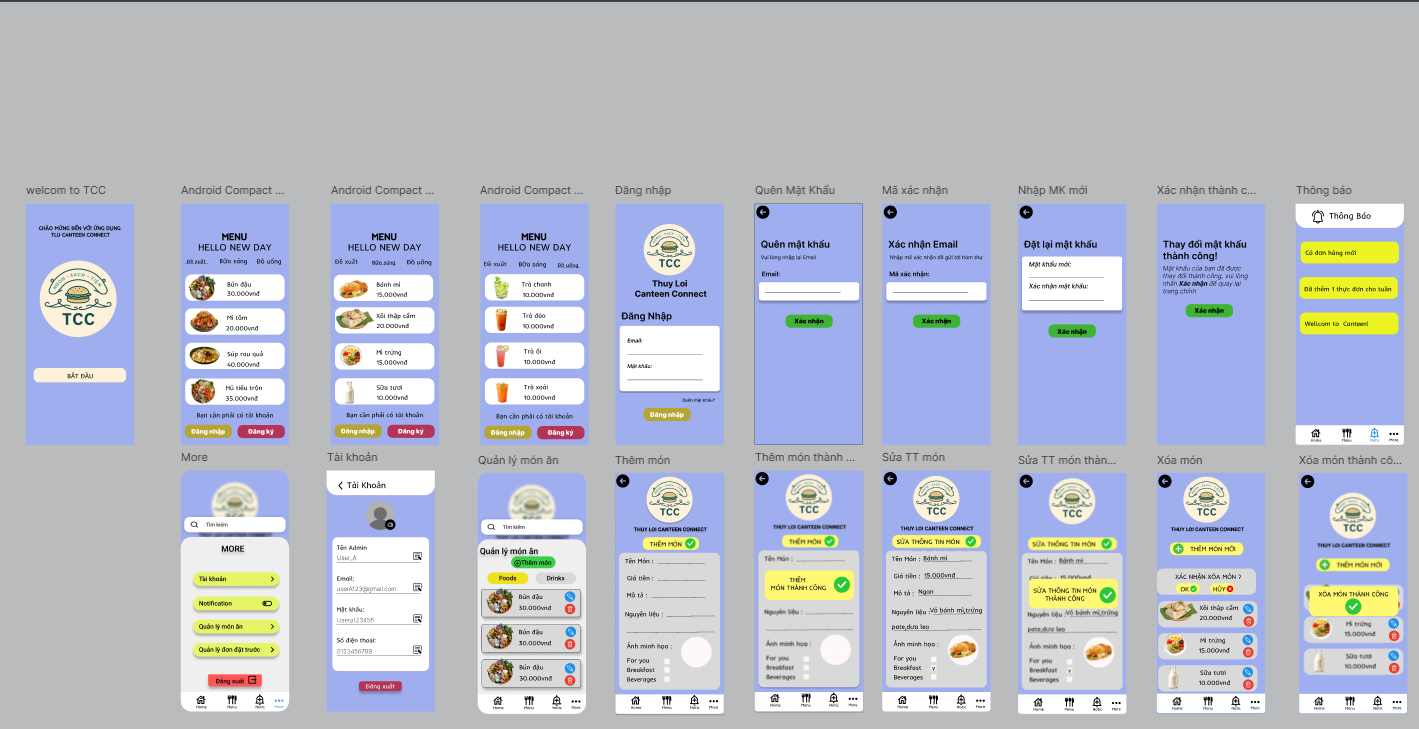
# **Chương 3: Xây dựng ứng dụng**

## **3.1. Thiết kế Figma**

*Link Github:*

<https://github.com/Nguyen-Tuan-Tu/Android-programming-homework.git>

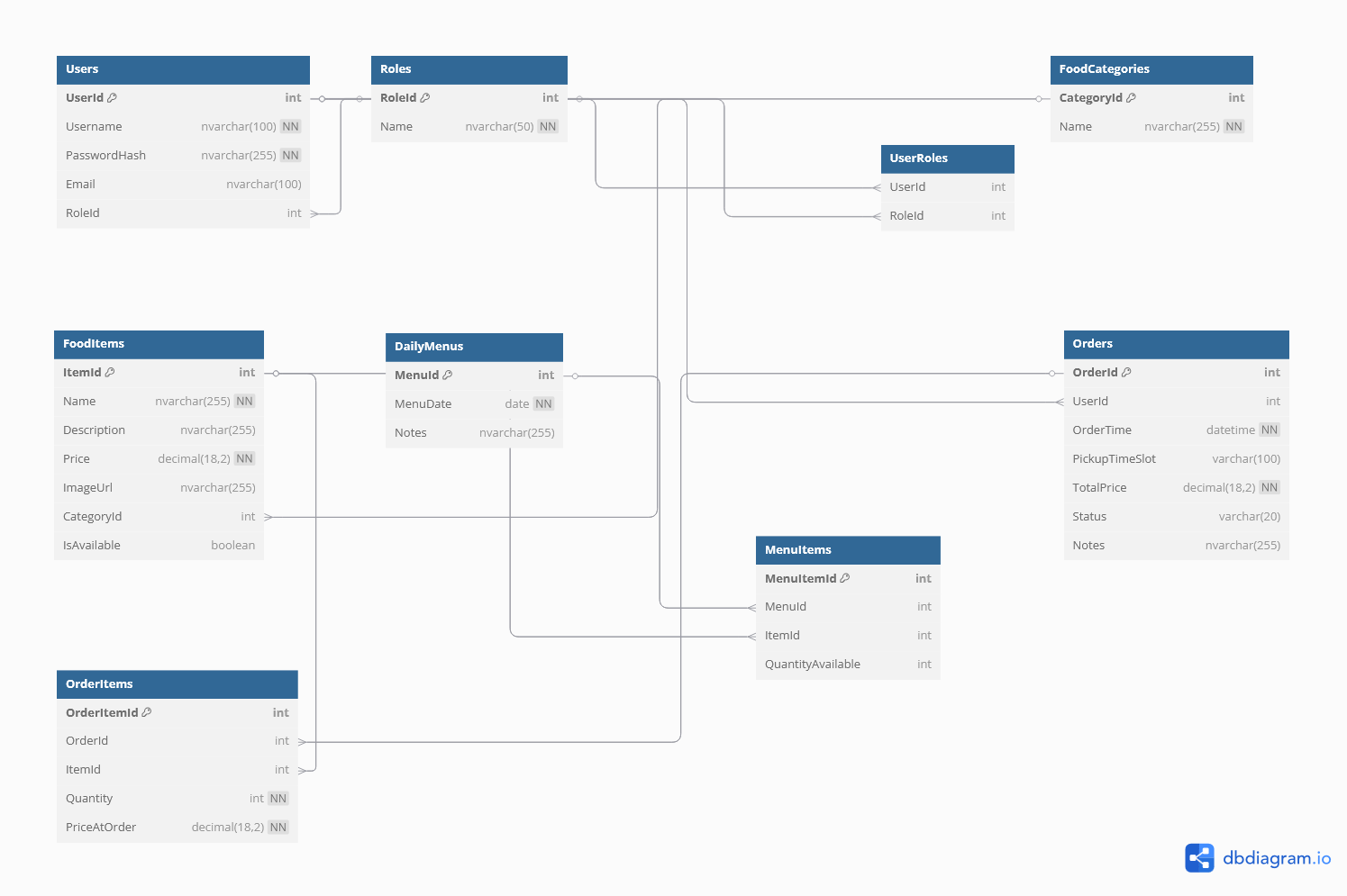
*Link Figma:* <https://www.figma.com/design/wWyNVXD8wga1ontPFRkeu4/Android-programming-homework?m=auto&t=zWM8CPS1JGFvElrq-6>







## **3.2.Thiết kế cơ sở dữ liệu**



**1. Bảng Users**

Đây là bảng chính quản lý người dùng trong hệ thống.

* UserId *(PK)*: Khóa chính, định danh từng người.
* Username, PasswordHash, Email: Thông tin cơ bản.
* RoleId: Khóa ngoại đến bảng Roles, dùng khi người dùng chỉ có một vai trò.

Ngoài ra, còn liên kết **nhiều-nhiều** qua bảng UserRoles.

**2. Bảng Roles**

* RoleId *(PK)*: Khóa chính.
* Name: Tên vai trò, ví dụ "Admin", "Student"...

Quan hệ:

* 1 vai trò có thể thuộc nhiều người dùng (UserRoles)
* 1 vai trò cũng có thể gắn trực tiếp cho 1 user (qua Users.RoleId)

**3. Bảng UserRoles (quan hệ N:N)**

Đây là bảng trung gian để tạo quan hệ **nhiều-nhiều** giữa Users và Roles.

* Gồm UserId, RoleId, cả 2 đều là FK và tạo thành PK kép.

**4. Bảng FoodCategories**

* CategoryId *(PK)*: Khóa chính.
* Name: Tên danh mục như “Món chính”, “Đồ uống”,...

Liên kết 1:N với bảng FoodItems (mỗi món ăn thuộc một danh mục).

**5. Bảng FoodItems (danh sách món ăn)**

* ItemId *(PK)*, tên món, mô tả, giá tiền, hình ảnh, trạng thái còn bán không (IsAvailable).
* CategoryId: FK đến bảng FoodCategories.

🔗 Mối quan hệ:

* 1 món ăn có thể xuất hiện trong nhiều thực đơn (MenuItems)
* 1 món có thể nằm trong nhiều đơn đặt món (OrderItems)

**6. Bảng DailyMenus (thực đơn theo ngày)**

* MenuId *(PK)*, ngày (MenuDate), ghi chú (Notes).

Mối quan hệ:

* 1 thực đơn có nhiều món ăn thông qua MenuItems.

**7. Bảng MenuItems**

* Liên kết giữa DailyMenus và FoodItems (nhiều món trong thực đơn).
* QuantityAvailable: số lượng món có sẵn hôm đó.

Mối quan hệ:

* 1 MenuId + 1 ItemId tạo thành cặp duy nhất.

**8. Bảng Orders (đơn đặt món)**

* OrderId *(PK)*, UserId *(FK)*, thời gian đặt, thời gian lấy (PickupTimeSlot), tổng tiền, trạng thái đơn (Status), ghi chú.

Trạng thái có thể là: *Placed, Confirmed, Preparing, Ready, Completed, Cancelled*

**9. Bảng OrderItems**

* Chi tiết từng món trong 1 đơn hàng.
* OrderItemId *(PK)*, OrderId, ItemId, số lượng, giá tại thời điểm đặt (PriceAtOrder).

**Các mối quan hệ nổi bật**

* **Users – Orders**: 1 user đặt nhiều đơn.
* **Orders – OrderItems – FoodItems**: đơn hàng có nhiều món.
* **DailyMenus – MenuItems – FoodItems**: thực đơn hàng ngày chứa nhiều món.
* **FoodItems – FoodCategories**: món thuộc một loại.

## **3.3.Giao diện người dùng**

3.3.1. Màn hình bắt đầu

|  |
| --- |
|  |

3.3.2.Màn hình trang chủ khi chưa đăng nhập/đăng ký

|  |  |  |
| --- | --- | --- |
|  |  |  |

3.3.3.Màn hình đăng ký

|  |  |  |
| --- | --- | --- |
|  |  |  |

3.3.4.Màn hình đăng nhập

|  |
| --- |
|  |

3.3.5.Màn hình đăng nhập khi quên mật khẩu

|  |  |
| --- | --- |
|  |  |
|  |  |

3.3.6.Màn hình trang chủ

|  |
| --- |
|  |

3.3.7.Màn hình hiện thị thực đơn (Đồ ăn/Đồ uống)

|  |  |
| --- | --- |
|  |  |

3.3.8.Màn hình hiển thị thông báo

|  |
| --- |
|  |

3.3.9.Màn hình hiện thị quản lý người dung (thông tin cá nhân, lịch sử đặt món)

|  |  |  |
| --- | --- | --- |
|  |  |  |

3.3.10.Màn hình giỏ hàng

|  |
| --- |
|  |

3.3.11. Màn hình chi tiết món ăn và thanh toán khi đặt trước

|  |  |
| --- | --- |
|  |  |

## **3.4.Giao diện quản trị viên**

3.4.1.Màn hình đăng nhập quản trị viên

|  |
| --- |
|  |

3.4.2.Màn hình trang chủ của quản trị viên

|  |
| --- |
|  |

3.4.3.Màn hình thông tin quản trị viên

|  |
| --- |
|  |

3.4.4.Màn hình quản lí món ăn

|  |
| --- |
|  |

3.4.5.Màn hình quản lý đơn đặt trước

|  |
| --- |
|  |

3.4.6.Màn hình xem danh sách đơn hàng

|  |  |  |
| --- | --- | --- |
|  |  |  |

3.4.7.Màn hình thống kê đơn hàng

|  |
| --- |
|  |

3.4.8.Màn hình quản lý danh sách thực đơn, chi tiết thực đơn

|  |  |
| --- | --- |
|  |  |

3.4.9.Màn hình Thêm, Cập nhật, Xoá thực đơn

|  |  |  |
| --- | --- | --- |
|  |  |  |

3.4.10.Màn hình thông báo

|  |
| --- |
|  |

## **3.5 Code minh hoạ các chức năng cốt lõi**

3.5.1.Đăng kí

1. **package** com.example.login; // Đảm bảo dòng này khớp với tên package của bạn
3. **import** android.content.Intent;
4. **import** android.os.Bundle;
5. **import** android.util.Patterns;
6. **import** android.view.View;
7. **import** android.widget.Button;
8. **import** android.widget.EditText;
9. **import** android.widget.ImageView; // Đã thêm ImageView
10. **import** android.widget.TextView;
11. **import** android.widget.Toast;
12. **import** androidx.appcompat.app.AppCompatActivity;
14. **public** **class** DangKiActivity **extends** AppCompatActivity {
16. **private** EditText etUsername;
17. **private** EditText etPassword;
18. **private** EditText etEmail;
19. **private** EditText etPhone;
20. **private** TextView tvEmailError;
21. **private** TextView tvPhoneError;
22. **private** Button btnRegister; // Đã đổi tên biến Register thành btnRegister để nhất quán
24. @Override
25. **protected** **void** onCreate(Bundle savedInstanceState) {
26. **super**.onCreate(savedInstanceState);
27. setContentView(R.layout.activity\_dang\_ki);
29. // Ánh xạ các View từ layout
30. etUsername = findViewById(R.id.etUsername);
31. etPassword = findViewById(R.id.etPassword);
32. etEmail = findViewById(R.id.etEmail);
33. etPhone = findViewById(R.id.etPhone);
34. tvEmailError = findViewById(R.id.tvEmailError);
35. tvPhoneError = findViewById(R.id.tvPhoneError);
36. btnRegister = findViewById(R.id.btnRegister); // Sử dụng lại biến đã khai báo
38. // Xử lý sự kiện click cho nút "Đăng ký"
39. // CHỈ GỌI registerUser() TỪ ĐÂY
40. btnRegister.setOnClickListener(**new** View.OnClickListener() {
41. @Override
42. **public** **void** onClick(View v) {
43. registerUser(); // Gọi phương thức kiểm tra validation và xử lý đăng ký
44. }
45. });
47. // Xử lý sự kiện click cho nút back
48. ImageView backButton = findViewById(R.id.ic\_back); // Đảm bảo ID này khớp với XML của bạn
49. **if** (backButton != **null**) {
50. backButton.setOnClickListener(**new** View.OnClickListener() {
51. @Override
52. **public** **void** onClick(View v) {
53. // Chuyển sang Activity Menu Đồ uống khi nhấn nút back (theo yêu cầu của bạn)
54. // Hoặc bạn có thể dùng onBackPressed(); để quay về màn hình trước đó
55. Intent intent = **new** Intent(DangKiActivity.**this**, Menu\_foryou\_Activity.**class**);
56. startActivity(intent);
57. finish(); // Tùy chọn: kết thúc Activity hiện tại nếu không muốn quay lại
58. }
59. });
60. }
62. // LOẠI BỎ ĐOẠN CODE BỊ TRÙNG LẶP NÀY
63. // Button Register = findViewById(R.id.btnRegister);
64. // Register.setOnClickListener(new View.OnClickListener() {
65. // @Override
66. // public void onClick(View v) {
67. // Intent intent = new Intent(DangKiActivity.this, Login\_Activity.class);
68. // startActivity(intent);
69. // }
70. // });
72. // LOẠI BỎ ĐOẠN CODE NÀY NẾU backButton ĐÃ XỬ LÝ Ở TRÊN
73. // ImageView Back = findViewById(R.id.ic\_back); // ID này không khớp với XML bạn đã cho
74. // Back.setOnClickListener(new View.OnClickListener() {
75. //     @Override
76. //     public void onClick(View v) {
77. //         Intent intent = new Intent(DangKiActivity.this, Menu\_foryou\_Activity.class);
78. //         startActivity(intent);
79. //     }
80. // });
81. }
83. **private** **void** registerUser() {
84. String username = etUsername.getText().toString().trim();
85. String password = etPassword.getText().toString().trim();
86. String email = etEmail.getText().toString().trim();
87. String phone = etPhone.getText().toString().trim();
89. **boolean** isValid = **true**; // Biến cờ để kiểm tra tổng thể
91. // Ẩn tất cả các thông báo lỗi trước khi kiểm tra lại
92. tvEmailError.setVisibility(View.GONE);
93. tvPhoneError.setVisibility(View.GONE);
94. etUsername.setError(**null**); // Xóa lỗi cũ
95. etPassword.setError(**null**); // Xóa lỗi cũ
97. // Kiểm tra validation từng trường
98. **if** (username.isEmpty()) {
99. etUsername.setError("Tên người dùng không được để trống");
100. isValid = **false**;
101. }
102. **if** (password.isEmpty()) {
103. etPassword.setError("Mật khẩu không được để trống");
104. isValid = **false**;
105. }
106. // Bạn có thể thêm các kiểm tra độ dài, độ mạnh mật khẩu ở đây
107. // if (password.length() < 6) {
108. //     etPassword.setError("Mật khẩu phải có ít nhất 6 ký tự");
109. //     isValid = false;
110. // }
112. // Kiểm tra email
113. **if** (email.isEmpty()) {
114. tvEmailError.setText("\*email không được để trống");
115. tvEmailError.setVisibility(View.VISIBLE);
116. isValid = **false**;
117. } **else** **if** (!Patterns.EMAIL\_ADDRESS.matcher(email).matches()) {
118. tvEmailError.setText("\*email không hợp lệ");
119. tvEmailError.setVisibility(View.VISIBLE);
120. isValid = **false**;
121. }
123. // Kiểm tra số điện thoại (ví dụ: bắt đầu bằng 0 và có 10 chữ số)
124. **if** (phone.isEmpty()) {
125. tvPhoneError.setText("\*số điện thoại không được để trống");
126. tvPhoneError.setVisibility(View.VISIBLE);
127. isValid = **false**;
128. } **else** **if** (!phone.matches("^0[0-9]{9}$")) {
129. tvPhoneError.setText("\*số điện thoại không hợp lệ");
130. tvPhoneError.setVisibility(View.VISIBLE);
131. isValid = **false**;
132. }

135. **if** (isValid) {
136. // NẾU TẤT CẢ THÔNG TIN HỢP LỆ, THÌ MỚI CHUYỂN SANG GIAO DIỆN LOGIN
137. Toast.makeText(**this**, "Đăng ký thành công! Đang chuyển đến màn hình đăng nhập...", Toast.LENGTH\_LONG).show();
139. // Chuyển sang Login\_Activity
140. Intent intent = **new** Intent(DangKiActivity.**this**, AdminLoginActivity.**class**);
141. startActivity(intent);
142. finish(); // Kết thúc DangKiActivity để người dùng không thể quay lại màn hình đăng ký sau khi đã đăng ký thành công
143. } **else** {
144. // Nếu có lỗi, hiển thị thông báo chung
145. Toast.makeText(**this**, "Vui lòng kiểm tra lại thông tin đăng ký", Toast.LENGTH\_LONG).show();
146. }
147. }
148. }

3.5.2 Đăng nhập

1. **package** com.example.login;
3. **import** android.content.Intent;
4. **import** android.os.Bundle;
5. **import** android.view.View;
6. **import** android.widget.Button;
7. **import** android.widget.EditText;
8. **import** android.widget.Toast;
10. **import** androidx.appcompat.app.AppCompatActivity;
12. **import** com.google.firebase.database.DataSnapshot;
13. **import** com.google.firebase.database.DatabaseError;
14. **import** com.google.firebase.database.DatabaseReference;
15. **import** com.google.firebase.database.FirebaseDatabase;
16. **import** com.google.firebase.database.ValueEventListener;
18. **public** **class** AdminLoginActivity **extends** AppCompatActivity {
20. **private** EditText emailEditText, passwordEditText;
21. **private** Button loginButton, forgotPasswordButton;
23. **private** FirebaseDatabase database;
24. **private** DatabaseReference usersRef;
26. @Override
27. **protected** **void** onCreate(Bundle savedInstanceState) {
28. **super**.onCreate(savedInstanceState);
29. setContentView(R.layout.activity\_admin\_login);
31. // Khởi tạo Firebase Realtime Database
32. database = FirebaseDatabase.getInstance();
33. usersRef = database.getReference("users");
35. // Ánh xạ các EditText và Button
36. emailEditText = findViewById(R.id.email);
37. passwordEditText = findViewById(R.id.password);
38. loginButton = findViewById(R.id.login\_button);
39. forgotPasswordButton = findViewById(R.id.forgotPasswordButton);
41. // Sự kiện khi nhấn nút Đăng nhập
42. loginButton.setOnClickListener(**new** View.OnClickListener() {
43. @Override
44. **public** **void** onClick(View v) {
45. String email = emailEditText.getText().toString().trim();
46. String password = passwordEditText.getText().toString().trim();
48. **if** (email.isEmpty() || password.isEmpty()) {
49. Toast.makeText(AdminLoginActivity.**this**, "Vui lòng nhập đầy đủ thông tin.", Toast.LENGTH\_SHORT).show();
50. } **else** {
51. checkLogin(email, password);
52. }
53. }
54. });
56. // Sự kiện khi nhấn nút Quên mật khẩu
57. forgotPasswordButton.setOnClickListener(v -> {
58. Intent intent = **new** Intent(AdminLoginActivity.**this**, ForgotPasswordActivityAdmin.**class**);
59. startActivity(intent);
60. });
61. }
63. **private** **void** checkLogin(String email, String password) {
64. // Kiểm tra dữ liệu người dùng trong Firebase
65. usersRef.addListenerForSingleValueEvent(**new** ValueEventListener() {
66. @Override
67. **public** **void** onDataChange(DataSnapshot dataSnapshot) {
68. **boolean** isValidUser = **false**;
69. String userRole = ""; // Biến lưu vai trò người dùng
71. // Lặp qua tất cả người dùng trong Realtime Database
72. **for** (DataSnapshot userSnapshot : dataSnapshot.getChildren()) {
73. String dbEmail = userSnapshot.child("email").getValue(String.**class**);
74. String dbPassword = userSnapshot.child("password").getValue(String.**class**);
75. userRole = userSnapshot.child("role").getValue(String.**class**); // Lấy vai trò người dùng
77. // So sánh email và password
78. **if** (email.equals(dbEmail) && password.equals(dbPassword)) {
79. isValidUser = **true**;
80. **break**;
81. }
82. }
84. // Kiểm tra xem có người dùng hợp lệ không
85. **if** (isValidUser) {
86. Toast.makeText(AdminLoginActivity.**this**, "Đăng nhập thành công!", Toast.LENGTH\_SHORT).show();
88. // Chuyển đến màn hình chính (MainScreenActivity) tùy vào vai trò người dùng
89. Intent intent;
90. **if** ("admin".equals(userRole)) {
91. intent = **new** Intent(AdminLoginActivity.**this**, MainScreenActivityAdmin.**class**);
92. } **else** {
93. intent = **new** Intent(AdminLoginActivity.**this**, HomePage\_Activity.**class**);
94. }
95. startActivity(intent);
96. finish(); // Đóng Activity hiện tại để không quay lại màn hình đăng nhập
97. } **else** {
98. Toast.makeText(AdminLoginActivity.**this**, "Email hoặc mật khẩu không đúng.", Toast.LENGTH\_SHORT).show();
99. }
100. }
102. @Override
103. **public** **void** onCancelled(DatabaseError databaseError) {
104. Toast.makeText(AdminLoginActivity.**this**, "Lỗi khi kết nối đến Firebase.", Toast.LENGTH\_SHORT).show();
105. }
106. });
107. }
108. }

3.5.3 Đặt trước món ăn + Chi tiết món ăn

1. **package** com.example.login;
3. **import** android.content.Intent;
4. **import** android.os.Bundle;
5. **import** android.view.View;
6. **import** android.widget.Button;
7. **import** android.widget.ImageView;
8. **import** android.widget.TextView;
9. **import** androidx.appcompat.app.AppCompatActivity;
10. **import** androidx.core.content.ContextCompat; // Import this
12. **public** **class** OrderActivity **extends** AppCompatActivity {
14. // Components from topBar
15. **private** ImageView topBarBackButton;
16. **private** TextView topBarTitle;
18. // Components from bottomNavBar
19. **private** ImageView navHome, navList, navBell, navCart;
21. // Components from activity\_order.xml
22. **private** TextView btnMinusThitBo, tvQuantityThitBo, btnPlusThitBo;
23. **private** TextView btnMinusKhoaiTay, tvQuantityKhoaiTay, btnPlusKhoaiTay;
24. **private** TextView btnMinusPhoMai, tvQuantityPhoMai, btnPlusPhoMai;
25. **private** TextView btnMinusTotalQuantity, tvTotalQuantity, btnPlusTotalQuantity;
26. **private** TextView tvTotalPrice;
27. **private** Button btnAddtoCart;
29. // Prices and counts
30. **private** **double** basePrice = 150000; // Giá Burger bò
31. **private** **double** priceThitBo = 15000;
32. **private** **double** priceKhoaiTay = 10000;
33. **private** **double** pricePhoMai = 8000;
35. **private** **int** quantityThitBo = 0;
36. **private** **int** quantityKhoaiTay = 0;
37. **private** **int** quantityPhoMai = 0;
38. **private** **int** totalQuantity = 1; // Số lượng burger tổng cộng ban đầu

41. @Override
42. **protected** **void** onCreate(Bundle savedInstanceState) {
43. **super**.onCreate(savedInstanceState);
44. setContentView(R.layout.activity\_order);
46. // Ánh xạ các thành phần từ top bar
47. topBarBackButton = findViewById(R.id.btn\_back);
48. topBarTitle = findViewById(R.id.toolbarTitle);
50. // Set tiêu đề cho màn hình "Đặt món"
51. topBarTitle.setText("Đặt món");
52. // Xử lý sự kiện click cho nút back
53. topBarBackButton.setOnClickListener(**new** View.OnClickListener() {
54. @Override
55. **public** **void** onClick(View v) {
56. onBackPressed(); // Quay về Activity trước đó (MainActivity)
57. }
58. });
60. // Ánh xạ các thành phần từ bottom nav bar
61. navHome = findViewById(R.id.nav\_home);
62. navList = findViewById(R.id.nav\_list);
63. navBell = findViewById(R.id.nav\_bell);
64. navCart = findViewById(R.id.nav\_cart);
66. // Highlight icon giỏ hàng vì đây là màn hình đặt món (tiếp theo của giỏ hàng)
67. navCart.setColorFilter(ContextCompat.getColor(**this**, R.color.primary\_app\_color)); // Dùng màu mới

70. // Ánh xạ các thành phần điều khiển số lượng và tổng tiền
71. btnMinusThitBo = findViewById(R.id.btnMinusThitBo);
72. tvQuantityThitBo = findViewById(R.id.tvQuantityThitBo);
73. btnPlusThitBo = findViewById(R.id.btnPlusThitBo);
75. btnMinusKhoaiTay = findViewById(R.id.btnMinusKhoaiTay);
76. tvQuantityKhoaiTay = findViewById(R.id.tvQuantityKhoaiTay);
77. btnPlusKhoaiTay = findViewById(R.id.btnPlusKhoaiTay);
79. btnMinusPhoMai = findViewById(R.id.btnMinusPhoMai);
80. tvQuantityPhoMai = findViewById(R.id.tvQuantityPhoMai);
81. btnPlusPhoMai = findViewById(R.id.btnPlusPhoMai);
83. btnMinusTotalQuantity = findViewById(R.id.btnMinusTotalQuantity);
84. tvTotalQuantity = findViewById(R.id.tvTotalQuantity);
85. btnPlusTotalQuantity = findViewById(R.id.btnPlusTotalQuantity);
87. tvTotalPrice = findViewById(R.id.totalPrice);
88. btnAddtoCart = findViewById(R.id.btnAddtoCart);
90. // Set giá trị ban đầu
91. tvQuantityThitBo.setText(String.valueOf(quantityThitBo));
92. tvQuantityKhoaiTay.setText(String.valueOf(quantityKhoaiTay));
93. tvQuantityPhoMai.setText(String.valueOf(quantityPhoMai));
94. tvTotalQuantity.setText(String.valueOf(totalQuantity));
95. updateTotalPrice();
97. // Setup listeners cho các nút tăng/giảm số lượng topping và số lượng tổng
98. setupQuantityControls(btnMinusThitBo, btnPlusThitBo, tvQuantityThitBo, "thitbo");
99. setupQuantityControls(btnMinusKhoaiTay, btnPlusKhoaiTay, tvQuantityKhoaiTay, "khoaitay");
100. setupQuantityControls(btnMinusPhoMai, btnPlusPhoMai, tvQuantityPhoMai, "phomai");
101. setupQuantityControls(btnMinusTotalQuantity, btnPlusTotalQuantity, tvTotalQuantity, "total");

104. // Xử lý sự kiện click cho nút "Thêm vào giỏ hàng"
105. btnAddtoCart.setOnClickListener(**new** View.OnClickListener() {
106. @Override
107. **public** **void** onClick(View v) {
108. // Tùy chọn: Chuyển đến PaymentActivity như trước, hoặc quay lại MainActivity (Giỏ hàng)
109. // Hiện tại giữ nguyên chuyển đến PaymentActivity
110. Intent intent = **new** Intent(OrderActivity.**this**, PaymentActivity.**class**);
111. startActivity(intent);
112. }
113. });
115. // Xử lý sự kiện click cho các icon ở bottom navigation bar (tùy chọn)
116. navHome.setOnClickListener(v -> {
117. // Đặt màu cho icon được chọn và reset màu cho các icon khác
118. navHome.setColorFilter(ContextCompat.getColor(**this**, R.color.primary\_app\_color));
119. navList.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
120. navBell.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
121. navCart.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
122. // Intent to HomeActivity
123. });
124. navList.setOnClickListener(v -> {
125. navHome.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
126. navList.setColorFilter(ContextCompat.getColor(**this**, R.color.primary\_app\_color));
127. navBell.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
128. navCart.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
129. // Intent to ListActivity
130. });
131. navBell.setOnClickListener(v -> {
132. navHome.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
133. navList.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
134. navBell.setColorFilter(ContextCompat.getColor(**this**, R.color.primary\_app\_color));
135. navCart.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
136. // Intent to NotificationActivity
137. });
138. navCart.setOnClickListener(v -> {
139. navHome.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
140. navList.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
141. navBell.setColorFilter(ContextCompat.getColor(**this**, R.color.dark\_gray\_text));
142. navCart.setColorFilter(ContextCompat.getColor(**this**, R.color.primary\_app\_color));
143. // Intent to CartActivity (not needed here if already on cart screen)
144. });
145. }
147. // Phương thức chung để xử lý tăng/giảm số lượng
148. **private** **void** setupQuantityControls(TextView minusBtn, TextView plusBtn, TextView countTv, String type) {
149. minusBtn.setOnClickListener(v -> {
150. **int** currentCount = Integer.parseInt(countTv.getText().toString());
151. **if** (currentCount > 0) {
152. currentCount--;
153. countTv.setText(String.valueOf(currentCount));
154. updateCounts(type, currentCount);
155. }
156. });
158. plusBtn.setOnClickListener(v -> {
159. **int** currentCount = Integer.parseInt(countTv.getText().toString());
160. currentCount++;
161. countTv.setText(String.valueOf(currentCount));
162. updateCounts(type, currentCount);
163. });
164. }
166. // Cập nhật biến số lượng và gọi tính lại tổng tiền
167. **private** **void** updateCounts(String type, **int** newCount) {
168. **switch** (type) {
169. **case** "thitbo":
170. quantityThitBo = newCount;
171. **break**;
172. **case** "khoaitay":
173. quantityKhoaiTay = newCount;
174. **break**;
175. **case** "phomai":
176. quantityPhoMai = newCount;
177. **break**;
178. **case** "total":
179. totalQuantity = newCount;
180. **if** (totalQuantity <= 0) totalQuantity = 1; // Đảm bảo số lượng tổng luôn ít nhất là 1
181. **break**;
182. }
183. updateTotalPrice();
184. }
186. // Tính toán và cập nhật tổng tiền
187. **private** **void** updateTotalPrice() {
188. **double** currentTotal = (basePrice +
189. (quantityThitBo \* priceThitBo) +
190. (quantityKhoaiTay \* priceKhoaiTay) +
191. (quantityPhoMai \* pricePhoMai)) \* totalQuantity;
193. tvTotalPrice.setText(String.format("%,.0f vnđ", currentTotal));
194. }
195. }

3.5.4 Tạo thực đơn

1. **package** com.example.login;
3. **import** android.content.Intent;
4. **import** android.os.Bundle;
5. **import** android.widget.Toast;
7. **import** androidx.appcompat.app.AppCompatActivity;
8. **import** androidx.recyclerview.widget.LinearLayoutManager;
9. **import** androidx.recyclerview.widget.RecyclerView;
11. **import** com.google.firebase.database.DataSnapshot;
12. **import** com.google.firebase.database.DatabaseError;
13. **import** com.google.firebase.database.DatabaseReference;
14. **import** com.google.firebase.database.FirebaseDatabase;
15. **import** com.google.firebase.database.ValueEventListener;
17. **import** java.util.ArrayList;
18. **import** java.util.List;
20. **public** **class** MenuActivityAdmin **extends** AppCompatActivity {
22. **private** RecyclerView recyclerView;
23. **private** MenuAdapter menuAdapter;
24. **private** List<Menu> menuList;
25. **private** DatabaseReference menuRef;
27. @Override
28. **protected** **void** onCreate(Bundle savedInstanceState) {
29. **super**.onCreate(savedInstanceState);
30. setContentView(R.layout.activity\_menu\_admin);
32. // Khởi tạo RecyclerView và adapter
33. recyclerView = findViewById(R.id.recycler\_view);
34. recyclerView.setLayoutManager(**new** LinearLayoutManager(**this**));
36. menuList = **new** ArrayList<>();
37. menuAdapter = **new** MenuAdapter(menuList);
38. recyclerView.setAdapter(menuAdapter);
40. // Khởi tạo Firebase Database reference
41. menuRef = FirebaseDatabase.getInstance().getReference("menu");
43. // Lấy dữ liệu từ Firebase lần đầu tiên khi Activity được tạo
44. loadMenuData();
46. // Xử lý sự kiện khi nhấn nút "Thêm thực đơn"
47. findViewById(R.id.btn\_add\_menu).setOnClickListener(v -> {
48. // Chuyển đến Activity thêm thực đơn
49. Intent intent = **new** Intent(MenuActivityAdmin.**this**, AddMenuActivityAdmin.**class**);
50. startActivity(intent);
51. });
52. }
54. @Override
55. **protected** **void** onResume() {
56. **super**.onResume();
57. // Lấy lại dữ liệu khi Activity quay lại từ AddMenuActivity
58. loadMenuData();
59. }
61. // Phương thức để lấy và cập nhật dữ liệu từ Firebase
62. **private** **void** loadMenuData() {
63. menuRef.addValueEventListener(**new** ValueEventListener() {
64. @Override
65. **public** **void** onDataChange(DataSnapshot dataSnapshot) {
66. menuList.clear();
67. // Lặp qua các ngày trong tuần và lấy thực đơn
68. **for** (DataSnapshot daySnapshot : dataSnapshot.getChildren()) {
69. String day = daySnapshot.getKey();  // "monday", "tuesday", ...
70. String name = daySnapshot.child("name").getValue(String.**class**);
72. List<String> food = **new** ArrayList<>();
73. **for** (DataSnapshot foodSnapshot : daySnapshot.child("food").getChildren()) {
74. food.add(foodSnapshot.getValue(String.**class**));
75. }
77. List<String> notes = **new** ArrayList<>();
78. **for** (DataSnapshot noteSnapshot : daySnapshot.child("notes").getChildren()) {
79. notes.add(noteSnapshot.getValue(String.**class**));
80. }
82. // Tạo đối tượng Menu và thêm vào list
83. Menu menu = **new** Menu(day, name, food, notes);
84. menuList.add(menu);
85. }
86. menuAdapter.notifyDataSetChanged(); // Cập nhật giao diện
87. }
89. @Override
90. **public** **void** onCancelled(DatabaseError databaseError) {
91. // Xử lý lỗi nếu có
92. Toast.makeText(MenuActivityAdmin.**this**, "Lỗi khi tải thực đơn!", Toast.LENGTH\_SHORT).show();
93. }
94. });
95. }
96. }

**KẾT LUẬN**

**1. Kết quả đạt được**

* **Ứng dụng hoạt động hiệu quả**: Dự án đã xây dựng thành công một ứng dụng giúp kết nối sinh viên và cán bộ với nhà ăn, giảm thiểu thời gian chờ đợi và cung cấp thông tin thực đơn dễ dàng. Sinh viên có thể đặt món trước, giúp việc chuẩn bị đồ ăn cho nhà ăn trở nên tối ưu hơn.
* **Tính năng đặt món trước**: Chức năng này giúp sinh viên tiết kiệm thời gian, tránh tình trạng hết món vào giờ cao điểm.
* **Quản lý nhà ăn hiệu quả**: Quản lý nhà ăn có thể theo dõi các đơn đặt trước và cập nhật thực đơn nhanh chóng.

**2. Nhược điểm**

* **Hạn chế về khả năng mở rộng**: Nếu trường có thêm nhiều nhà ăn hoặc cơ sở khác, hệ thống hiện tại có thể gặp khó khăn trong việc mở rộng, vì hệ thống không tính đến nhiều nhà ăn hoặc cơ sở ở các vị trí khác.
* **Chưa hỗ trợ đa nền tảng hoàn toàn**: Dự án hiện tại mới chỉ nhắm tới ứng dụng Web hoặc Android, chưa có sự hỗ trợ đầy đủ cho các hệ điều hành khác như iOS.
* **Chưa thực hiện kiểm thử đầy đủ**: Việc kiểm thử và thử nghiệm có thể chưa đầy đủ, gây ra một số lỗi hoặc tình huống ngoài mong đợi mà chưa được xử lý tối ưu.

**3. Hướng phát triển**

* **Mở rộng tính năng và hỗ trợ nhiều nhà ăn**: Phát triển thêm tính năng hỗ trợ nhiều nhà ăn hoặc các cơ sở khác, giúp ứng dụng có thể phục vụ rộng rãi hơn trong trường hợp mở rộng quy mô.
* **Cải tiến giao diện và trải nghiệm người dùng (UX/UI)**: Giao diện người dùng có thể được tối ưu hóa hơn nữa, đặc biệt là đối với những người chưa quen với công nghệ. Thêm tính năng như thanh toán qua các phương thức mới sẽ tạo thuận lợi hơn cho người dùng.
* **Đa nền tảng**: Mở rộng ứng dụng để hỗ trợ cả trên iOS, giúp tiếp cận nhiều đối tượng hơn.
* **Tích hợp AI và dữ liệu lớn**: Áp dụng các công nghệ AI để dự đoán nhu cầu món ăn theo xu hướng tiêu dùng, giúp quản lý nhà ăn tối ưu hóa lượng nguyên liệu cần chuẩn bị.

Tóm lại, "TLU Canteen Connect" là một ứng dụng hữu ích giúp giải quyết các vấn đề thực tế trong nhà ăn của trường Đại học Thủy Lợi, nhưng vẫn có thể phát triển thêm về khả năng mở rộng và tối ưu hóa trải nghiệm người dùng.