APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) 1600-1618
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Aliahj Dlicious Dessert Online Ordering Application

Nur Liyana Sufri[1], Rozlini Mohamed[1]*

[1] *Faculty of Computer Science and Information Technology,*
Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: rozlini@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.086

Article Info

Received: 11 June 2025
Accepted: 18 June 2026
Available online: 30 June 2026

Keywords

e-Commerce, online ordering, dessert, mobile application

Abstract

Aliahj Dlicious Bakery is a local shop that offers a variety of desserts and food items, including pastries, cakes, and heavy meals. Currently, the shop manages customer orders manually using online platforms such as WhatsApp, Facebook, and Instagram, with orders recorded in notebooks and Excel sheets. This manual approach leads to several issues, including missed orders, slow processing, and limited engagement with customers. To overcome these challenges, the Aliahj Dlicious Dessert Online Ordering Application was developed. The goal of the application is to improve efficienc and enhance customer satisfaction. Developed using Flutter and Firebase, the application provides features such as online ordering, menu browsing, booking, delivery tracking, and feedback submission. Implementation of the application improves order accuracy, minimizes manual errors, and provides a seamless and user-friendly experience. Customers are now able to place orders anytime without visiting the shop physically. Future upgrades may include AI-based recommendations to personalize the user experience.

1. Introduction

"Aliahj Dlicious Bakery is a dessert shop" located in the Parit Yaani area, sells a variety of desserts and cake. The shop only offers in-store purchases and manual ordering which is using WhatsApp for special event orders. With the increasing use of online platforms, businesses are shifting from traditional physical operations to digital environments. Many businesses are now leveraging the growing popularity of online food ordering due to its convenience, speed, accuracy, and transparency [1]. Online food ordering systems allow customers to browse items, compare options, read reviews, and place orders anytime, improving customer relationships [2]. For a dessert shop, having an online platform presents a valuable opportunity for growth and business expansion. However, the current operation of Aliahj Dlicious Bakery faces several challenges. The shop still relies on a manual ordering system, where orders are written down in a physical book. Desserts and cakes are promoted through online platforms like WhatsApp, Instagram, and Facebook. This method lacks real-time synchronization with the actual stock of the shop and operations. When certain desserts become unavailable, customers are not immediately informed. To place an order, customers must send a message to the shop, and those who wish to view the full menu need to visit the store physically, which is inconvenient and time-consuming. Furthermore, updates regarding the availability of desserts, delivery services, and booking options are not consistently accessible to all customers because only regular customers tend to stay informed.

To resolve this problem, an Aliahj Dlicious Dessert Online Ordering Application is developed. This system will feature an interactive menu page that showcasing all available desserts, enabling customers to browse easily. A shopping cart and checkout process will allow customers to place orders online, enhancing convenience and reducing the need for in-person visits. Additionally, the system will incorporate a reservation module, enabling customers to book orders for special occasions directly through the website. Real-time updates about daily

specials will be communicated through the site, increasing visibility and engagement. By implementing this online system, the dessert shop can significantly improve its operational efficiency, enhance customer satisfaction, and expand its market reach. Consequently, transitioning to an online ordering system could streamline the process, making it more efficient and accessible for both the shop and its customers [3].

The report begins with an introduction that outlines the background, problem statement, objectives, scope, expected outcomes, applied methodology, project timeline, and overall significance of the system. The next section reviews relevant theoretical frameworks and literature to provide a strong foundation for system design and development. The methodology adopted in this study follows a prototyping model, comprising six iterative phases: requirement gathering, quick design, prototype development, user evaluation, refinement, and implementation. The system analysis and design section explain the logical structure of the system using modeling tools such as Use Case diagrams, UML diagrams, Entity Relationship Diagrams (ERD), and database normalization techniques. The implementation and testing section presents the practical development of the system and evaluates its functionality through various testing methods. The final section concludes the paper by summarizing key findings, identifying limitations, and proposing directions for future enhancements.

2. Literature Review

2.1 Comparison with the Existing Systems

During the development of the proposed system, several existing systems were analyzed for comparison. The selected systems for comparison in this project are Secret Recipe [4], Eat Cake Today [5], and Cake Together [6]. Table 1 presents the comparison between these three existing systems and the proposed system.

Table 1 *System Comparison*

Features/System	Secret Recipe	Eat Cake Today	Cake Together	Aliahj Dlicious Dessert
Log in / Register	✓	✓	✓	✓
Membership	✓	×	×	✓
Category / Search	✓	✓	✓	✓
Menu	✓	✓	✓	✓
Cart	✓	✓	✓	✓
Delivery	×	✓	✓	✓
Order History	✓	×	×	✓
Menu Description	×	✓	✓	✓
Customization	×	✓	✓	✓

The system comparison highlights the features offered by Secret Recipe, Eat Cake Today, Cake Together, and Aliahj Dlicious Dessert. All four systems provide essential features like login and registration, category or search functionality, menu display, and a shopping cart. However, only Aliahj Dlicious Dessert and Secret Recipe offer a membership feature, which Eat Cake Today and Cake Together lack. Delivery services are available in Aliahj Dlicious Dessert, Cake Together, and Eat Cake Today, but not in Secret Recipe. Secret Recipe and Aliahj Dlicious Dessert include order history functionality, while Cake Together and Eat Cake Today do not. Menu descriptions are provided by Cake Together, Eat Cake Today, and Aliahj Dlicious Dessert, but this feature is missing in Secret Recipe. Lastly, customization options are available only in Aliahj Delicious Dessert and Cake Together, whereas Secret Recipe and Eat Cake Today do not support this feature.

3. Methodology

This chapter explains the methodology used for the project, covering the selected model, its phases, and the workflow involved in system development. The approach adopted for this project is the prototyping model.

3.1 Prototyping Model

Prototyping can be defined as the Conversion of an intangible idea either related to the physical or digital world into a tangible configuration to test its feasibility, validity or efficacy [7]. The Prototyping model is created, tested, and refined until a satisfactory version is achieved [8]. This approach serves as a foundation for developing the final system or software. It is particularly useful in situations where project requirements are not fully defined or understood. The model is iterative, relying on a process of trial and error between the developer and client [8]. Figure 1 illustrates the prototyping model phases.

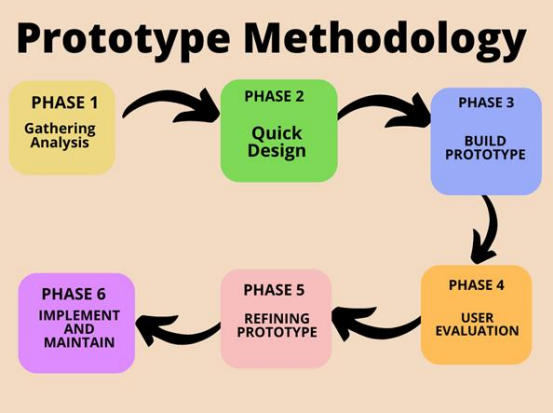


Fig. 1 *Prototyping Model Phases*

3.2 System Development Workflow

There are six phases from the prototype model as shown in figure 1. As shown in Table 2, each designed with specific assignments and deliverables to be accomplished during the development process. Every phase focuses on producing targeted outputs that contribute to the overall progression of the project.

Table 2 *Software development activities and their tasks*

Phase	Task	Output
Requirement Gathering	○ Identify user need ○ Conduct interviews with the shop manager ○ Document system requirement	○ Interview notes ○ Documented requirement ○ Flowchart
Quick Design	○ Sketch interface ○ Create low-fidelity wireframes ○ Outline core features	○ DFD & ERD ○ System Architecture ○ Database Schema & Data Dictionary ○ User Interface Design
Build Prototype	○ Code basic functionalities ○ Develop clickable prototype ○ Set up database	○ Functional prototype ○ Initial coded of the system
User Evaluation	○ Conduct a usability testing ○ Gathering user feedback ○ Analyze interactions	○ Usability test result ○ User Feedback summaries
Refinement	○ Modify design and functionality based on the feedback	○ Enhanced prototype ○ Updated features ○ Revised the system workflows
Implement and maintain	○ Finalize code ○ Deploy system ○ Post-launch the application	○ Functional system ○ Deployment documentation ○ Book manual of the application

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1600-1618

3.3 System Requirements Functional and Non-Functional requirements tables

Functional requirements define the specific operations, features, and services that the system must perform to meet user needs [3]. These include the core functionalities that ensure the system operates as intended. Table 3 shows the functional requirements of the proposed system.

Table 3 *Functional Requirements Tables*

No	Module	Description
1	User Management Module	• Allow the new users to register new account before login. • Allow the existing users to login with the id and password. • Alerts users for invalid login attempts. • Redirects users to the main menu after successful login.
2	Menu Management Module	• Allow the administrator to update the daily menu details, including images, name of the menu and prices. • Ensures real-time reflection of menu changes.
3	Order Module	• The system should provide customers with product details, including images, descriptions, and prices. • The system should enable adding items to the shopping cart. • The system should support modification of cart quantities. • The system should offer multiple payment options, such as credit/debit cards, digital wallets, and cash on delivery.
4	Booking Module	• The system should enable customers to book cakes or desserts for specific occasions. • The system should ensure proper scheduling and management of reservations.
5	Review and Ratings Module	• The system should allow customers to submit feedback and rate purchased items. • The system should provide guidance to potential customers based on reviews.
6	Delivery Module	• The system should manage the delivery process, including scheduling and real-time tracking. • The system should provide delivery status updates to customers.
7	Notification Module	• The system should send notifications about order status and promotions.

On the other hand, non-functional requirements address the system performance attributes, including reliability, scalability, usability, and security [9]. These requirements collectively ensure the system meets both the operational and quality expectations of users and stakeholders. Table 4 presents the non-functional requirements of the developed system.

Table 4 *Non – Functional Requirements*

No	Requirement	Description
1	Performance	The app should work smoothly when users browse the menu, make bookings, or place orders, even when many people use it at the same time.
2	Operational	Pages like the menu, cart, and checkout should load quickly, around 3 to 5 seconds when internet connection is stable.
3	Security	The app should keep user data safe, especially login details, order history, and booking information, using Firebase Authentication.
4	Cultural and political	The system should support both English and Bahasa Malaysia. It should avoid sensitive content and follow common privacy rules in Malaysia.

3.4 System Analysis

System design describes the overall structure or flow of the system, including its function [10]. the design of the application is explained. Object-oriented approach is used to generate the UML diagrams, which are Use Case Diagram, Activity Diagram, Sequence Diagram and Class Diagram.

3.4.1 Use Case Diagram

A UML Use Case Diagram represents the essential system requirements for a software application by visualizing the interaction between users that referred as actors and the system functions that referred as use cases. It helps identify how different types of users interact with the system and what actions are available to them. Figure 2 illustrates the Use Case Diagram for the Aliahj Dlicious Dessert Online Ordering Application. In the figure 2, two main actors are shown as Customer and Admin. The Customer can perform actions such as register, log in, view menu, place order, make booking, track delivery, submit feedback, and receive notifications. The admin can manage menu, process orders and deliveries, handle bookings, and monitor feedback and notification functions. Each use case is displayed as an oval and linked to the relevant actor, showing which user can access which feature. This diagram provides a clear overview of core functions and user interactions within the application and serves as a reference for further development and design.

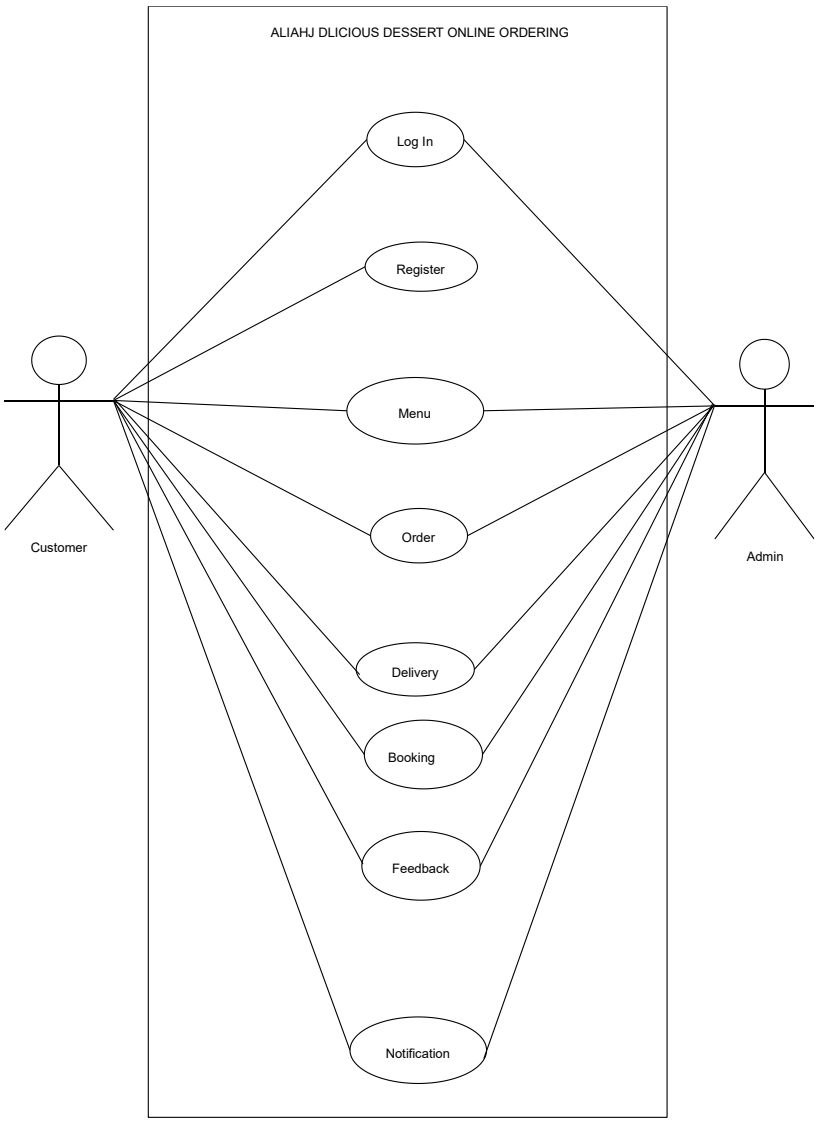


Fig. 2 *Use Case Diagram*

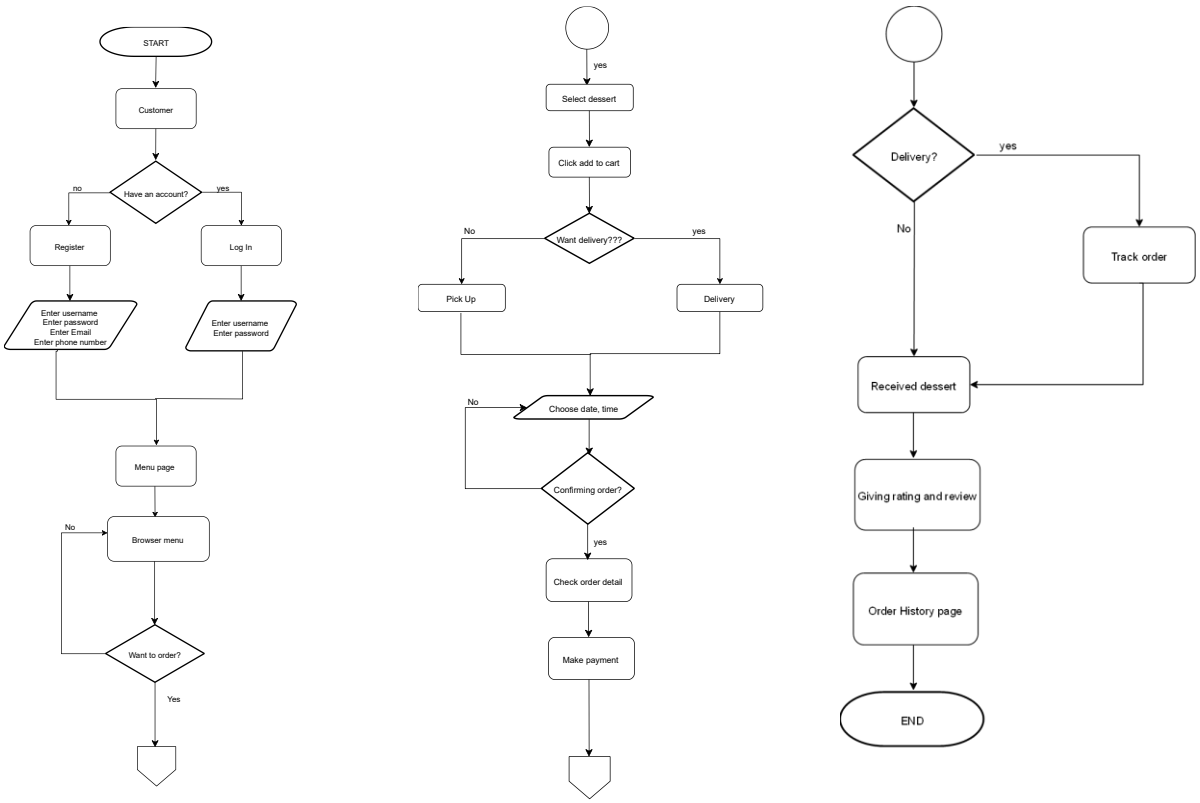
3.4.2 Erd

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1600-1618

In Appendix A, the Entity Relationship Diagram (ERD) shows the structure of the database used in the Aliahj Dlicious Dessert Online Ordering Application. An ERD is used to represent how the system data is organized and how different components are connected. The ERD includes seven main tables. The customer table stores user information such as name, contact number, and address. It uses customer_id as the primary key. The menu table contains details about dessert items, including menu_id, image, name, price, and description. The order table records customer orders and links each order to a customer and a menu item through customer_id and menu_id as foreign keys. This table also includes order date, status, quantity, and total amount. The delivery table manages delivery information such as date, time, and address, using delivery_id as the primary key and order_id as a foreign key. The feedback table stores reviews and ratings provided by customers. It includes feedback_id as the primary key and connects to order_id.

There are also two tables for notifications in Appendix A. The dev_Notification table handles delivery-related messages with timestamps, while the pay_Notification table stores updates about payment status and methods. Both tables use customer_id, and dev_Notification also uses order_id as a foreign key. This ERD defines the structure and relationships within the database. It supports important system features such as order handling, delivery tracking, feedback collection, and notification management.

3.4.3 Flowchart



(a) **(b)** **(c)**

Fig. 3 *Flowchart for Customer (a) Flowchart start; (b) Flowchart continued; (c) Flowchart ended*

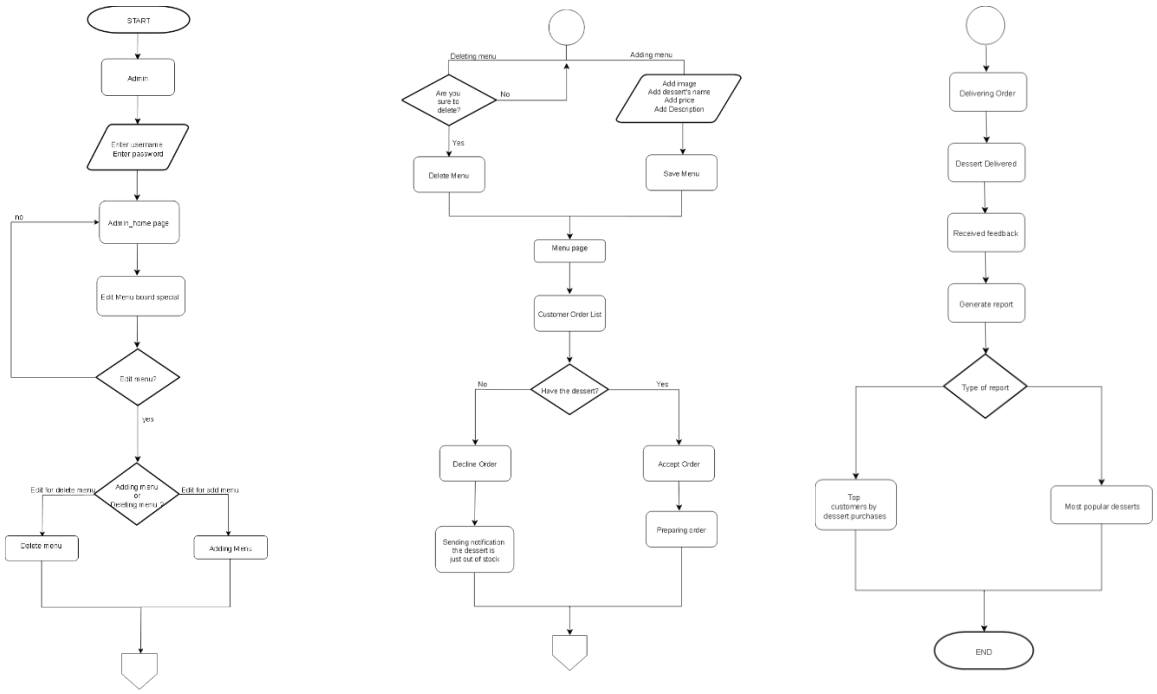


Fig. 4 *Flowchart for Admin (a) Flowchart start; (b) Flowchart continued; (c) Flowchart ended*

Figure 3 shows the customer flowchart diagrams of using the Aliahj Dlicious Dessert Online Ordering Application. In Figure 3(a), the flow begins with the login or registration process. If the user is not yet registered, the system will guide them to complete the registration first. Once logged in successfully, the user is directed to the menu page. Figure 3(b) continues with the customer browsing the menu, selecting desserts, and adding items to the cart. The process then proceeds to checkout, where the customer selects a payment method and places the order. After the order is submitted, the system updates the order status automatically. In Figure 3(c), the final part of the customer journey includes tracking the delivery, viewing order history, and submitting feedback. These flowcharts provide a clear overview of how a customer interacts with the application from start to finish.

Meanwhile, Figure 4 illustrates the admin flowchart diagrams that explain how the administrator manages the system. In Figure 4(a), the process starts with the admin login, which leads to the admin home page. From there, as shown in Figure 4(b), the admin can manage menu items by adding, updating, or deleting them. Figure 4(c) illustrates the admin reviewing customer orders, approving or rejecting them based on stock, and updating delivery statuses. Additionally, the admin can view event bookings and manage customer feedback. These flowcharts demonstrate how each admin task is integrated into the system, ensuring smooth operation and effective management of the online ordering platform.

3.5 System Design

3.5.1 System architecture

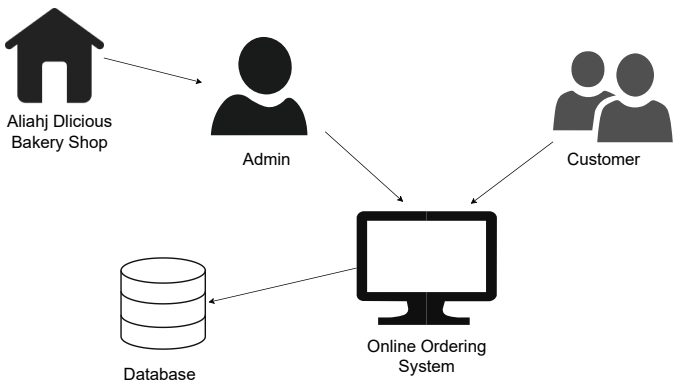


Fig. 5 *System Architecture*

Figure 5 provides a high-level view of the system and its interaction with external entities. It outlines the overall structure and data flow between the system and the users. The central system, labeled as the Aliahj Dlicious Dessert Online Ordering Application, connects with multiple external entities including Admin, Customer, and Database. The customer interacts with the system to perform actions such as browsing the menu, placing orders, and receiving notifications. The admin manages operations like updating menu items, processing orders, and viewing feedback. The database stores essential data including user information, order records, menu details, and transaction history. Additionally, the diagram shows home interface and system interface, representing user access points such as web or mobile platforms. This context diagram helps to clearly define the boundary of the system and the major interactions with external actors, serving as a foundation for understanding how data flows in and out of the system.

3.5.2 Relational Schema

The relational schema is represented through tables below that include:

i. `Customer (customer_id, customer_Name, customer_Password, customer_PhoneNum, customer_Email, customer_address)`

ii. `Menu (menu_id, menu_Image, menu_Name, menu_price, menu_description)`

iii. `Order (order_id, order_date, totalAmount, orderStatus, quantityOrder)`

iv. `Delivery (delivery_id, deliveryDate, deliveryTime, deliveryAddress)`

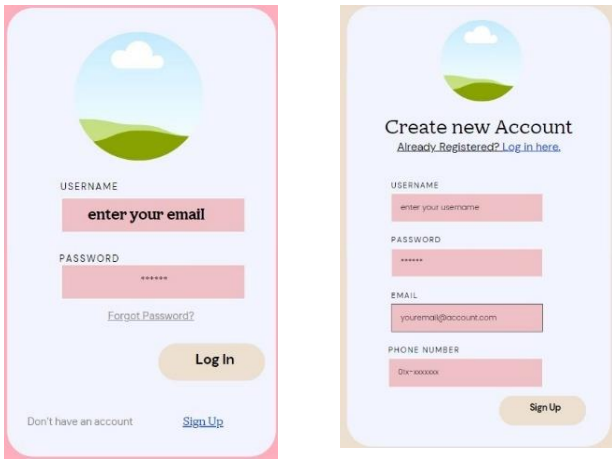
v. `Feedback (feedback_id, rating, reviewText)`

vi. `Dev_Notification (devNotification_id, message, sent_Date)`

vii. `Pay_Notification (payNotification_id, paymentStatus, paymentMethod, paymentDate)`

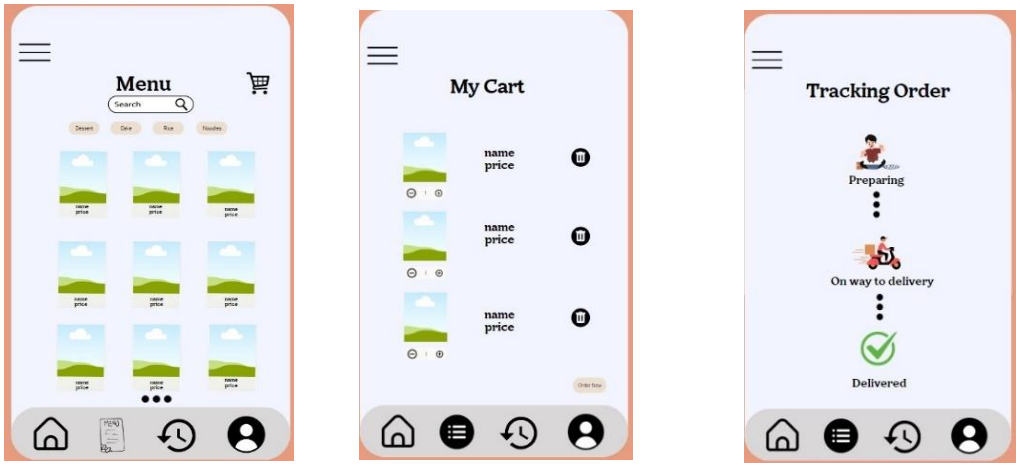
3.5.3 UI Design

The user interface design, which represents the flow of the application, is shown in Figures 6 and 7 using simple visual illustrations created with tools such as Canva. Figure 6 includes two screens: the login interface (a), which allows a user to enter email and password to access the system, and the registration interface (b), which provides input fields for username, password, email, and phone number to create a new account. Figure 7 displays three core screens in the application. The menu interface (a) shows a collection of dessert items with images, names, and prices arranged in a grid layout. The cart interface (b) lists selected items for confirmation before making a purchase. The delivery interface (c) shows real-time order tracking through three stages, which are Preparing, On way to delivery, and Delivered.



(a) **(b)**

Fig. 6 *User Interface (a) Login; (b) Register*



(a) **(b)** **(c)**

Fig. 7 *User Interface (a) Menu; (b) Cart; (c) Delivery*

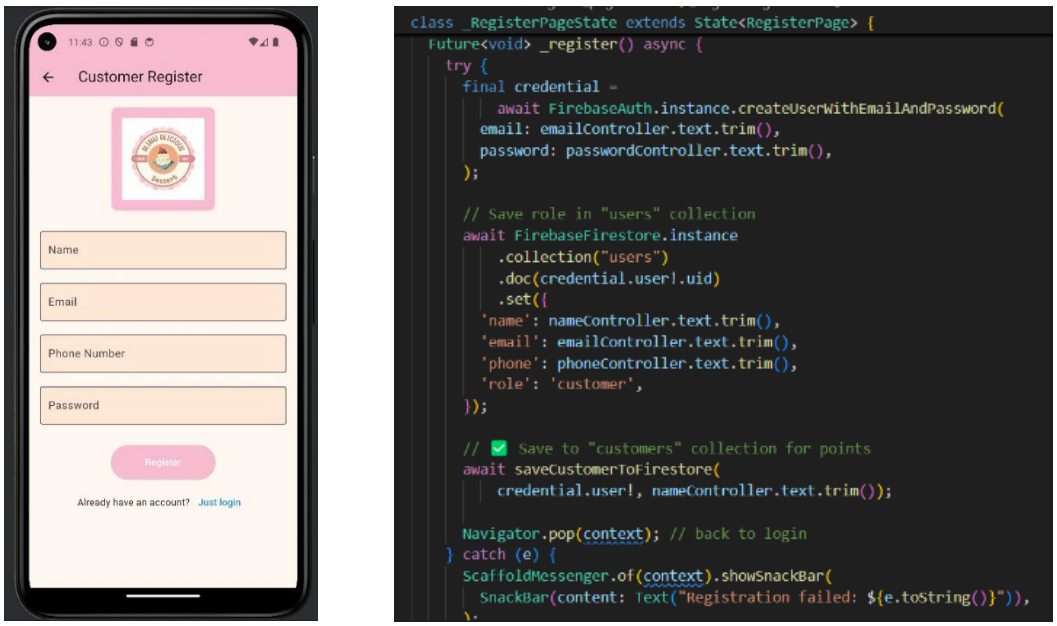
4. Result and Discussion

The Aliahj Dlicious Dessert Online Ordering Application was implemented using Flutter for frontend development with the Dart programming language, and Firebase as the backend database. The main development tools used were Visual Studio Code for coding and the Android Emulator for running the application interface.

4.1 Module

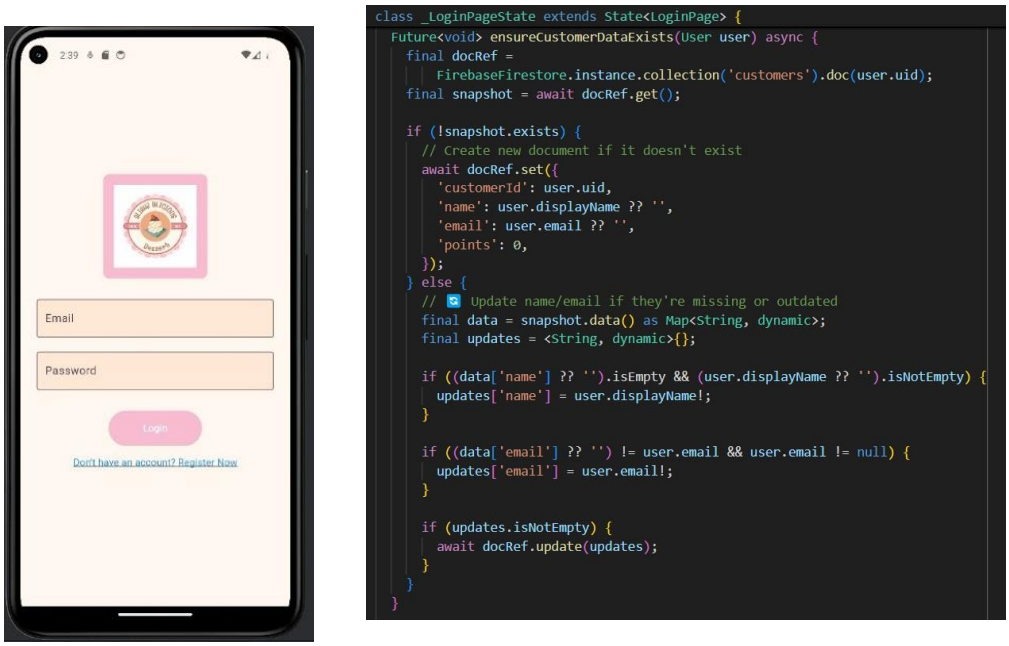
4.1.1 User Management Module

The user management module in this application allows new users, such as customers or newly hired staff, to register for an account if they do not already have one. Users must create an account before accessing the home page and exploring the application. To register, users are required to provide their email, password, confirm their password, and enter their phone number on the registration page. Once the registration is successful, the account is created, and users can log in using their email and password. A role-based security feature is implemented to ensure that customers cannot access the admin section, as unauthorized access is restricted.



(a) **(b)**

Fig. 8 *Register (a) Interface; (b) Source Code*



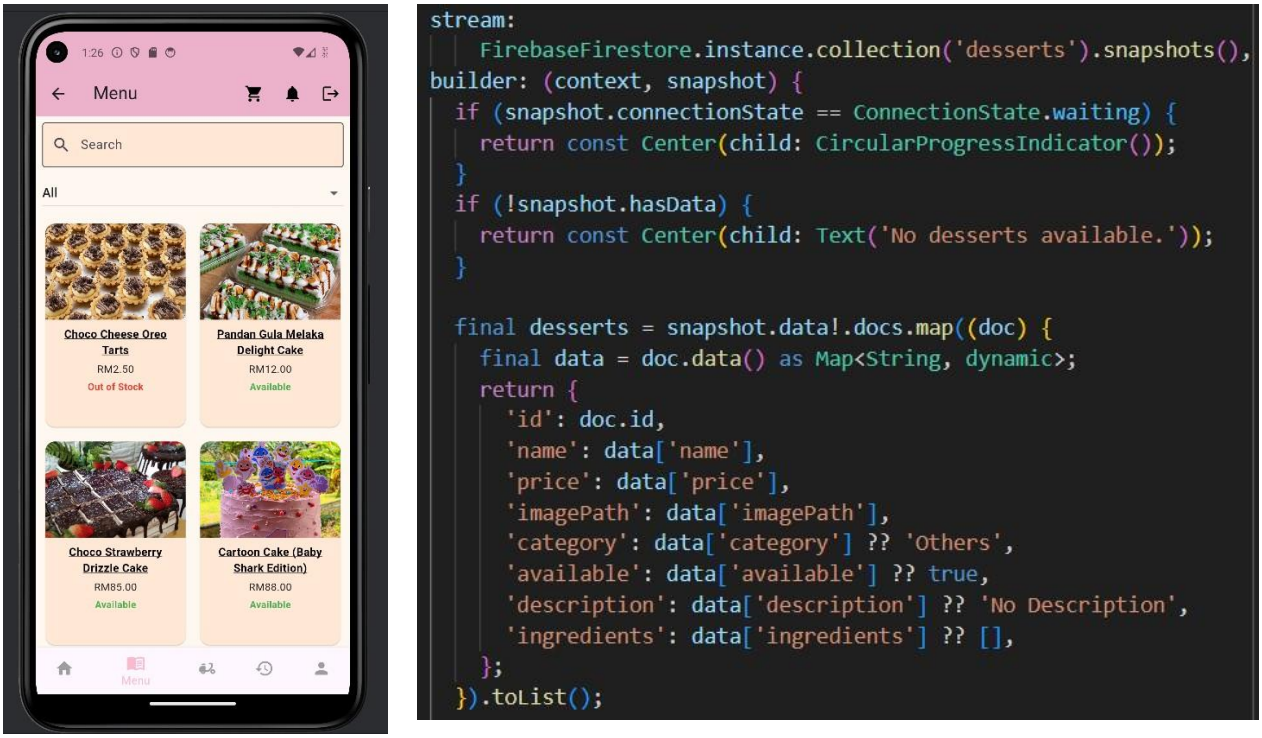
(a) **(b)**

Fig. 9 *Login (a) Interface; (b) Source Code*

Figure 8 presents the customer registration interface (a) and its corresponding source code (b). The interface includes fields for name, email, phone number, and password, along with a shop logo for branding. The source code shows the function that creates a new user account using Firebase Authentication and stores user details in the Firestore database with a defined role as "customer." Figure 9 displays the login interface (a) and the source code (b) used for customer authentication. The code checks the credentials and updates user data in Firestore if any changes are detected. A role-based access control feature is implemented to ensure that only authorized users can access specific parts of the system, such as the admin section.

4.1.2 Menu Management Module

In this application, the menu management feature allows the admin to manage the dessert menu. This includes adding new menu items, deleting existing ones, and editing details such as the menu name, price, description, and list of ingredients. The admin also manages the stock levels of desserts and meals. Additionally, the admin controls the 'Today's Menu,' which is align with the customer-facing 'Today's Menu' board.



(a) **(b)**

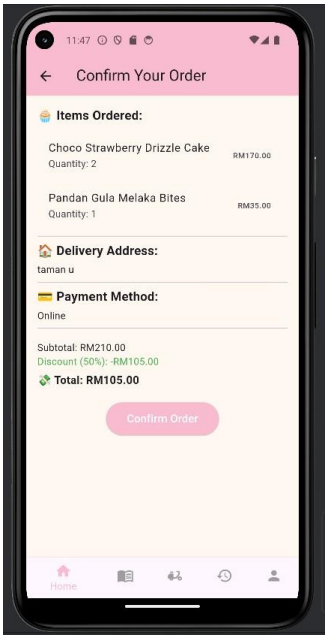
Fig. 10 *Menu (a) Interface; (b) Source Code*

Figure 10 shows the menu interface (a) and the related source code (b). Based on the interface, some menu items are marked as out of stock, while others are available for ordering. The layout displays dessert images, names, prices, and stock status in a clear format. The source code retrieves data from the Firestore collection named "desserts" and renders each item with details such as image, name, price, availability, description, and ingredients. This implementation allows the customer menu page to display menu data dynamically based on the current database content.

4.1.3 Order Module

In this module, the customer order feature allows users to add items to the cart, choose a payment method, and view a list of desserts being purchased. It also includes the delivery address and the total price of the order. Meanwhile, on the admin side, the order management feature allows the admin to view customer orders and either accept or reject them, especially if an item is suddenly out of stock. The admin can also view detailed customer information along with the specifics of each dessert ordered.

Figure 11 shows the customer interface (a) for placing orders, with a clear layout for selecting menu items, choosing a delivery method, and proceeding with the order. The source code (b) demonstrates the implementation used to handle customer orders. It captures relevant order details such as user ID, order date, payment method, delivery address, total amount, and a list of ordered items. This functionality enables the system to process transactions and store order records in the database effectively.



(a)

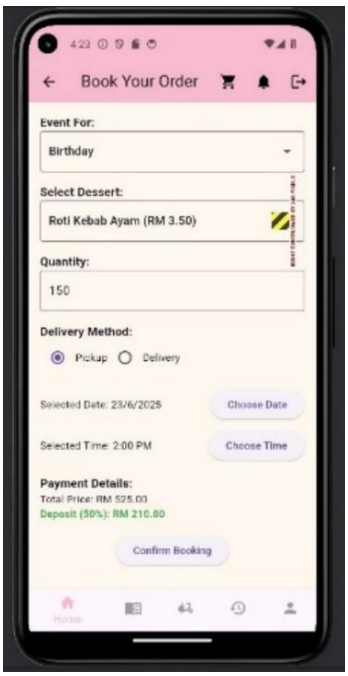
```
// Save the order
await FirebaseFirestore.instance
    .collection('orders')
    .doc(orderId)
    .set({
  'userId': user.uid,
  'userName': user.email ?? 'Unknown',
  'orderDate': FieldValue.serverTimestamp(),
  'paymentMethod': paymentMethod,
  'deliveryAddress': deliveryAddress,
  'totalAmount': totalAmount,
  'discountApplied': discount,
  'finalAmount': finalAmount,
  'totalQuantity': totalQuantity,
  'desserts': orderItems
      .map((item) => {
            'name': item['name'],
            'quantity': item['quantity'],
            'price': item['price'],
          })
      .toList(),
});
```

(b)

Fig. 11 *Order (a) Interface; (b) Source Code*

4.1.4 Booking Module

In the booking module, customers can reserve desserts or other meals for special events. They have the option to choose either delivery or self-pickup. A deposit is required to confirm the booking, and the remaining payment will be made once the desserts are delivered or collected. On the admin side, the admin can view customer booking details, which helps in preparing the order and organizing for the event or occasion.



(a)

```
void submitBooking() async {
  if (_formKey.currentState!.validate() &&
      selectedDessert != null &&
      selectedEvent != null) {
    try {
      await FirebaseFirestore.instance.collection('bookings').add({
        'event': selectedEvent,
        'dessert': selectedDessert,
        'quantity': quantity,
        'deliveryMethod': deliveryMethod,
        'totalAmount': totalAmount,
        'depositAmount': depositAmount,
        'date': selectedDate?.toIso8601String(),
        'time': selectedTime?.format(context),
        'timestamp': FieldValue.serverTimestamp(),
      });

      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(
            "Booking confirmed! Deposit: RM ${depositAmount.toStringAsFixed(2)}"),
      )); // SnackBar
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Failed to save booking: $e")),
      );
    }
```

(b)

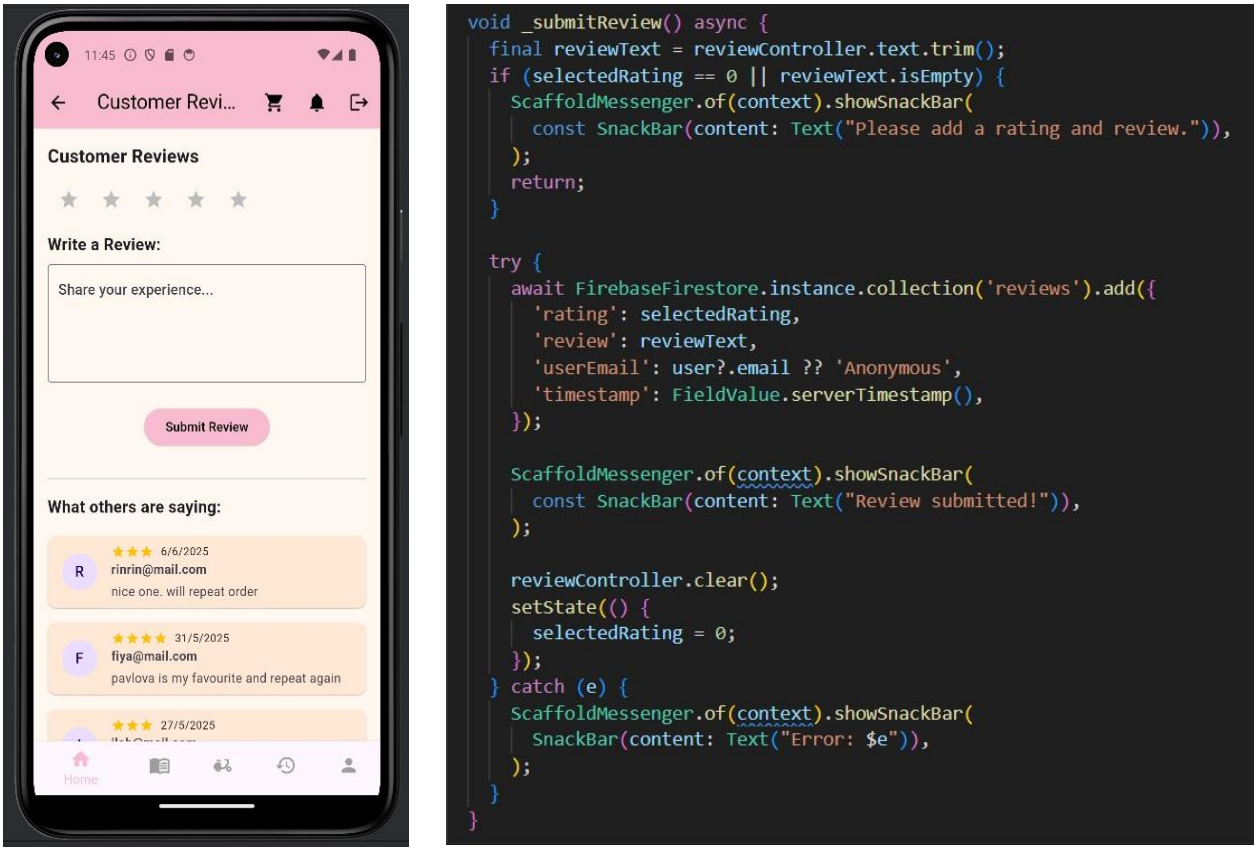
Fig. 12 *Booking (a) Interface; (b) Source Code*

Figure 12 shows the customer booking interface (a) and the corresponding source code (b). The interface includes options for selecting dessert, entering quantity, choosing a delivery method, setting a date and time, and specifying payment details. The source code demonstrates how booking data is stored in the Firestore database, including

selected dessert, quantity, method, date, and payment amount. This implementation ensures that all relevant booking information is captured accurately for processing by the system.

4.1.5 Review and Ratings Module

In the review and rating module, customers can rate the desserts using a star rating system and provide written feedback based on their purchase experience. Meanwhile, the admin can view all submitted ratings and reviews from customers, allowing them to monitor feedback and improve service quality.



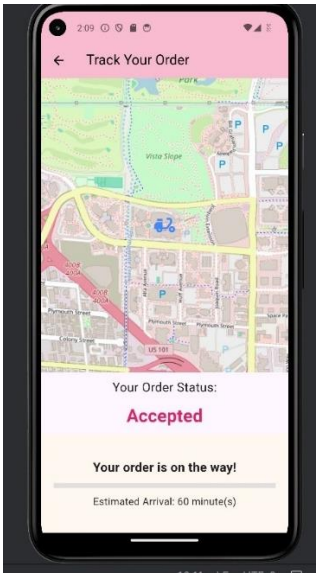
(a) **(b)**

Fig. 13 *Review and Rating (a) Interface; (b) Source Code*

Figure 13 displays the customer review and rating interface (a) along with the source code (b). The interface allows users to provide feedback by submitting written reviews and selecting star ratings based on their experience. The source code handles the process of capturing input, storing feedback data in the Firestore collection named "reviews," and confirming successful submission. This module enables the system to collect and display customer feedback effectively for future reference and quality improvement.

4.1.6 Delivery Module

In the delivery module, the customer and admin interfaces are quite similar, with slight differences in functionality. Customers can track the status of their order, including where it is and the estimated time of arrival. On the other hand, the admin can monitor whether an order has been successfully delivered to the customer or is still in progress.



(a)

```dart
Future<void> simulateDriverMovement() async {
  try {
    final driverRef =
        FirebaseFirestore.instance.collection('drivers').doc('driver123');

    double lat = 1.9550;
    double lng = 103.0285;

    for (int i = 0; i < 30; i++) {
      await Future.delayed(const Duration(seconds: 2));
      lat += 0.0001;
      lng += 0.0001;

      await driverRef.update({'lat': lat, 'lng': lng});
    }
  } catch (e) {
    debugPrint('Simulation error: $e');
  }
}
```

(b)

Fig. 14 *Delivery (a) Interface; (b) Source Code*

Figure 14 shows the delivery interface (a) and the related source code (b). The interface allows the customer to view the status of an order along a visual map, providing updates such as "Accepted" and estimated delivery progress. The source code simulates driver movement by updating latitude and longitude values in the Firestore database at timed intervals. This implementation enables real-time delivery tracking, allowing the customer to monitor order progress throughout the delivery process.

4.1.7 Notification Module

The notification module was tested using Firestore to ensure that both admin and customer users can receive notifications in real time.

```dart
Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  await Firebase.initializeApp();
  print('📩 Background message: ${message.messageId}');
}

Run | Debug | Profile
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  try {
    // Initialize Firebase
    await Firebase.initializeApp(
      options: DefaultFirebaseOptions.currentPlatform,
    );

    FirebaseMessaging messaging = FirebaseMessaging.instance;

    // Request Web FCM token using VAPID key
    if (kIsWeb) {
      String? token = await messaging.getToken(
        vapidKey:
            "BLNBjuLeBq_sHpjU3IQ0ilMkH4IAxjtMDtrc4JU4SB4QSM8eRKtJw_kWEel5hifPDmlrZ-5BTDRfuOVrOO8HOVA",
      );

      print('🔥 FCM Token: $token');
    }

    // Request notification permissions
    NotificationSettings settings = await messaging.requestPermission();
    debugPrint('🔐 Permission granted: ${settings.authorizationStatus}');

    // Foreground notification handler
    FirebaseMessaging.onMessage.listen((RemoteMessage message) {
      final notification = message.notification;
      final context = navigatorKey.currentContext;
```

Fig. 15 *Notification Source Code*

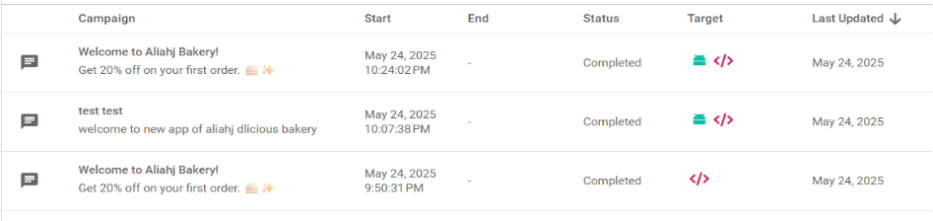


Fig. 16 *Notification Store in the database*

Figure 15 shows the source code for setting up Firebase Messaging, including initialization, token generation, permission requests, and message handling. Figure 16 displays the database storing notification records with details like content, status, target, and delivery time, allowing efficient message tracking.

4.2 Testing Result

System testing was performed to verify that the developed system fulfills both functional and non-functional requirements. The testing process is considered complete when all modules operate correctly according to the defined specifications. Table 5 presents the detailed results of the system testing. A total of 7 modules were tested to evaluate the system. The summary of the test case outcomes is shown in Table 5 below.

Table 5 *Testing Results*

Module	Description	Expected Result	Actual Results	Pass/Fail
User Management	To check whether users can register and login an account	Users able to create and log in to an account	Account created and login successful	Pass
Menu	To check whether admin can add, update, and delete menu	Menu is added, edited, and deleted successfully	Menu functions worked correctly	Pass
	To check whether customer can view the menu	Customer can view all available desserts	Menu is displayed properly on customer side	Pass
Order	To check whether admin can view and update order status	Order details and status are visible and editable	Orders shown and status updated by admin	Pass
	To check whether customer can place and view orders	Order placed with correct total and shown in list	Customer order recorded and displayed	Pass
Booking	To check whether admin can approve or reject booking requests	Booking status can be updated	Booking approved/rejected successfully	Pass
	To check whether customer can submit a booking	Booking form is submitted with required fields	Booking saved correctly in system	Pass
Review and Rating	To check whether admin can view submitted reviews	Reviews should be listed for admin view	Reviews shown properly on admin side	Pass
	To check whether customer can rate and review	Customer can submit rating and comment	Rating and review submitted successfully	Pass
Delivery	To check whether order location updates are shown	Delivery location updates in real time	Location shown correctly to both sides	Pass
Notification	To check whether notifications are sent and received	Messages delivered to target users	Notifications sent and received successfully	Pass

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1600-1618

4.2.1 4.User Acceptance Testing

The primary goal of user acceptance testing is to gather feedback and insights from actual users who will use the system to perform specific tasks. For this project, the bakery owner and a customer were selected as the testers. To support the testing process, a simple feedback form was provided for them to complete after using the system. Figures 30 display the results of the user acceptance testing conducted by the owner and the customer.

As shown in Figure 17 presents the results of user acceptance testing from the customer perspective. Most participants gave the highest ratings for features such as login, dessert menu presentation, cart and checkout process, reservation booking, order tracking, and notification alerts. This indicates that the system is user friendly and performs well in delivering a smooth customer experience.

Figure 18 shows the admin testing results. Admin users also gave positive feedback, especially on features such as menu updates, viewing customer information, managing order status, and receiving customer reviews. The majority agreed that the system is simple to use and supports efficient task management with clear notifications.

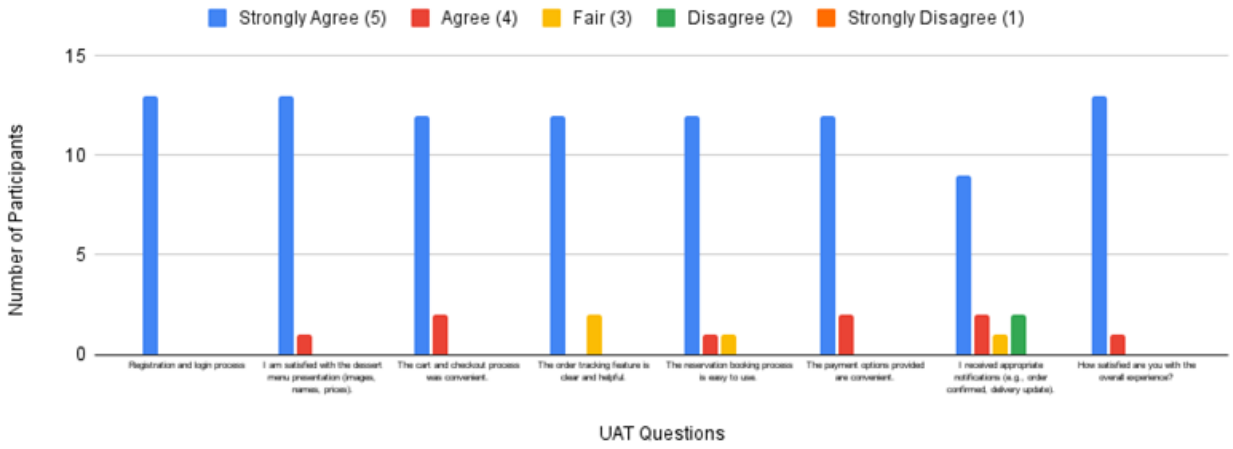


Fig. 17 *General User Acceptance Testing Customer*

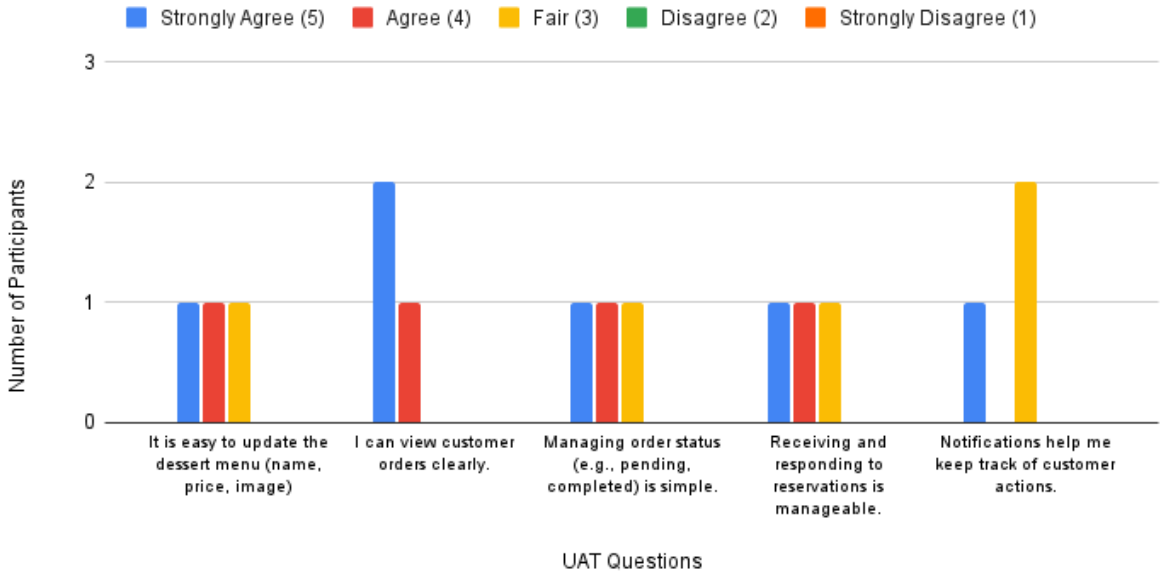


Fig. 18 *General User Acceptance Testing Admin*

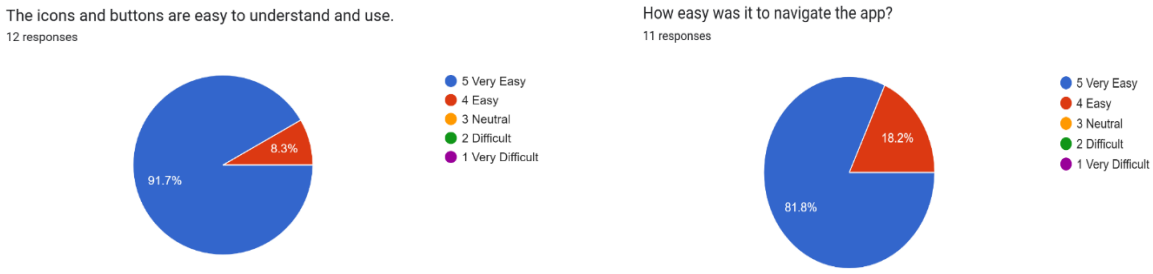


Fig. 19 *Chart Testing on Interface Usability and App Navigation*

Figure 19 presents the results related to the user interface and navigation experience. The majority of users (91.7%) found the icons and buttons very easy to understand and use, while a small percentage (8.3%) rated them as easy. Additionally, 81.8% of users reported that navigating the app was very easy, and 18.2% found it easy. These results indicate that the system interface is intuitive and user-friendly, contributing to a smooth and accessible user experience.

5. Conclusion

In conclusion, the development of the Aliahj Dlicious Dessert Online Ordering Application provides a digital solution to replace the manual order management process used by Aliahj Dlicious Bakery. This transition helps resolve challenges related to manual order tracking, missed orders, and inefficient operations. The system is designed to streamline and automate the processes of ordering, purchasing, and delivery, making them more effective and convenient for both the bakery and its customers. The application offers a user-friendly interface that enhances the customer experience while also improving overall operational efficiency. It enables customers to place orders at any time and from any location, increasing satisfaction and accessibility. By implementing this system, the bakery can overcome current limitations and is better prepared for future business expansion. Overall, the application delivers a modern and practical approach for handling dessert orders and supports the digital growth of Aliahj Dlicious Bakery in a competitive marketplace.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

There is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Nur Liyana Sufri, Rozlini Mohamed;* ***data collection:*** *Nur Liyana Sufri;* ***analysis and interpretation of results:*** *Nur Liyana Sufri, Rozlini Mohamed;* ***draft manuscript preparation:*** *Nur Liyana Sufri, Rozlini Mohamed. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] D. M. Luffy, "Online Ordering Systems: Development of Order Calculation using a Web-Based Calculator," Enhanced Knowledge in Sciences and Technology, vol. 3, no. 1, pp. 135–142, 2023.
[2] W. L. S. P. Jayawardena, Food Ordering & Management System, Doctoral dissertation, 2021.
[3] V. Chavan, P. Jadhav, S. Korde, and P. Teli, "Implementing customizable online food ordering system using web-based application," International Journal of Innovative Science, Engineering & Technology, vol. 2, no. 4, pp. 722–727, 2015.
[4] Secret Recipe, "Company Profile," [Online]. Available: https://www.secretrecipe.com.my/
[5] Eat Cake Today, "Online Cake Delivery Service," [Online]. Available: https://www.eatcaketoday.com/
[6] Cake Together, "Online Dessert Platform," [Online]. Available: https://caketogether.com/
[7] The Engineering Projects, "What is Prototyping? Meaning, Types, Process, Tools and Examples," May 04, 2021. [Online]. Available: https://www.theengineeringprojects.com
[8] U. S. Senarath, "Waterfall methodology, prototyping and agile development," Tech. Rep., pp. 1–16, 2021.
[9] Y. Yahiaoui, Firebase Cookbook: Over 70 Recipes to Help You Create Real-Time Web and Mobile Applications with Firebase. Birmingham, UK: Packt Publishing, 2017.
[10] C. Zaccagnino, Programming Flutter: Native, Cross-Platform Apps the Easy Way. The Pragmatic Programmers, LLC, 2020.

Appendix A: ERD