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

AITCS

Vol. 6 No. 2 (2025) 981-999
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

UTHM Bargain: Marketplace App

Wan Muhammad Aniq Irfan Wan Mohd Lutpi[1], Norhanim Selamat[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: norhanim@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.053

Article Info

Received: 18 July 2025
Accepted: 20 November 2025
Available online: 30 November 2025

Keywords

Mobile App Marketplace, Prototype Model. Secure Transaction

Abstract

The widespread use of messaging platforms like WhatsApp and Telegram for buying and selling goods among UTHM community members has led to challenges such as inefficient item browsing and vulnerability to fraud due to the lack of proper verification. This project aims to develop UTHM Bargain, a dedicated mobile application for secure trading exclusively among UTHM members, verified through matriculation numbers. Developed using the prototype model, the project followed an object-oriented approach and applied user interface design principles with secure authentication mechanisms. The software used includes Flutter for mobile development and Firebase for backend services. Testing results showed the app successfully addressed the initial issues by offering an organized marketplace and robust user verification. Future enhancements may include integration of secure payment systems and expansion to other institutions.

1. Introduction

In the digitalization of campus communities has transformed how students buy and sell items [1]. While platforms like WhatsApp and Telegram are commonly used, they pose challenges such as limited item organization, repeated postings, and security concerns [2]. At Universiti Tun Hussein Onn Malaysia (UTHM), the need for a secure, dedicated platform has become increasingly important. Studies show that authentication mechanisms like multi-factor verification can reduce fraud and enhance trust in digital transactions [3]. This research UTHM Bargain, a mobile application tailored for the UTHM community. The main objectives are: (i) to develop the system using object-oriented programming, (ii) to design a mobile app with user verification via matric numbers, and (iii) to conduct acceptance testing. The app aims to support secure secondhand trading within the UTHM community through strict user verification protocols. The expected outcome is a user-friendly platform that promotes sustainable item exchange while providing a safer alternative to general-purpose messaging apps [4].

2. Related Work

This section discuss the technology used to develop the UTHM Bargain mobile application and analyze three similar existing systems such as Carousell, Mudah.my, and Facebook Marketplace.

2.1 Technology Used

The development of UTHM Bargain utilizes Flutter for cross-platform mobile development and Dart as its programming language. Flutter enables the creation of native apps for both Android and iOS using a single codebase. Firebase supports backend functionality, including real-time databases, secure authentication, and cloud storage for product images. The project also uses Visual Studio Code as the IDE, Git for version control, and Material Design for a consistent user interface.

2.2 Study of Existing Related Systems

This section examines current marketplace platforms such as Carousell, Mudah.my, and Facebook Marketplace. Each offers features like user ratings, chat functions, or social integration but lacks institution-specific verification. Their limitations highlight the need for a secure, campus-focused solution like UTHM Bargain.

Fig. 1 (a) C*arousell Main Page; (b) Mudah.my Main Page; (c) Facebook Marketplace Home Page*

In examining the current digital marketplace landscape, Figure 1 shows three popular platforms serve as relevant benchmarks for UTHM Bargain: Carousell, Mudah.my, and Facebook Marketplace. Carousell, widely used in Southeast Asia, offers an intuitive interface, integrated chat, and a user-based rating system. However, while it supports basic user verification, its lack of institutional-level safeguards makes it unsuitable for university-specific trading environments [5]. Despite effective category-based navigation, its open structure leaves users vulnerable to fraud and unverified interactions. Mudah.my, Malaysia's largest classified platform, offers simple navigation and broad item categories accessible via both web and mobile. Its usability appeals to a wide user base, but interactions are often off-platform commonly redirected to WhatsApp—which raises concerns about transaction security and privacy [6]. The absence of strong verification protocols makes it inadequate for secure campus-based marketplaces. Facebook Marketplace benefits from social network integration, offering buyer-seller interactions tied to user profiles, which can enhance trust [7]. Nevertheless, it lacks dedicated fraud prevention systems and institution-specific features. The open access and general-purpose nature limit its suitability for controlled environments like universities, where verified identities and closed communities are essential. These comparisons highlight the need for a secure, dedicated platform tailored to educational institutions, where trust, verification, and community-focused design are essential.

2.3 Comparison with the Existing Systems

Table 1 presents a comprehensive comparison of key features between the three existing marketplace systems (Carousell, Mudah.my, and Facebook Marketplace) and the proposed UTHM Bargain application. This analysis reveals significant gaps in existing platforms' capabilities to serve specialized institutional needs, particularly highlighting how UTHM Bargain's targeted approach.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 981-999

Table 1 *Comparative Analysis of Three Existing Systems and the Proposed System*

Features	Carousell	Mudah.my	Facebook Marketplace	UTHM Bargain
Specific Community	No	No	No	Yes
Strict User Verification	Yes	No	No	Yes
In-App Messaging	Yes	Yes	Yes	Yes
Auction Feature	No	No	No	Yes
Supports Sustainable Practices	No	No	No	Yes
Fraud Prevention Measures	Yes	No	No	Yes
Mobile-Friendly Platform	Yes	Yes	Yes	Yes
Tailored Features for Users	No	No	No	Yes
Secure Transaction Environment	Yes	No	No	Yes

Table 1 presents a comparative analysis of four system, Carousell, Mudah.my, Facebook Marketplace, and UTHM Bargain, across several features. While all systems offer in-app messaging and mobile-friendly platforms, UTHM Bargain consistently stands out by providing features like specific community, auction features, support for sustainable practices, tailored features for users, and secure transaction environment, which are largely absent in the other three existing systems. Carousell offers strict user verification and fraud prevention measures, unlike Mudah.my and Facebook Marketplace, but still lags behind UTHM Bargain in comprehensive offerings.

3. Methodology

The Prototype Model was selected as the development methodology for the UTHM Bargain mobile application due to its iterative nature and emphasis on user involvement. This approach consists of six key phases: requirement gathering, quick design, prototype development, user evaluation, refinement, and final implementation. Each stage enables progressive enhancement of the system through continuous feedback, ensuring the application aligns with user expectations. The model's focus on rapid prototyping and frequent testing reduces development risks, improves usability, and ensures features such as item listings, messaging, and secure authentication are user-centric. Given its flexibility and structured feedback loops, the Prototype Model in Figure 2 is well-suited for delivering a reliable, student-focused trading platform within the university environment [8]. The detailed tasks and outcomes associated with each development phase are outlined in Table 2.

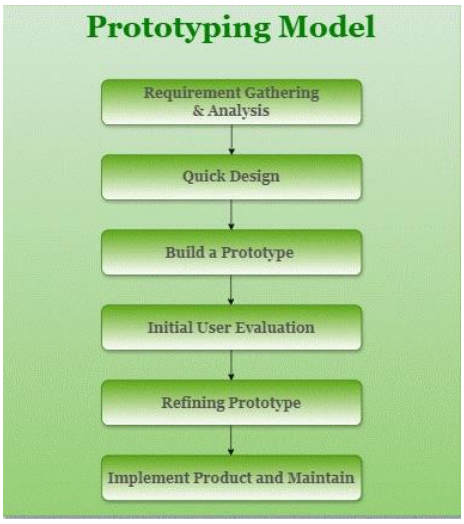


Fig. 2 *Prototype Model [8]*

Table 2 *System Development Phase Task and Deliverable*

Phase	Activity	Results
Requirements Gathering and Analysis	• Conduct interviews, surveys, and discussions with UTHM students to identify needs. • Define key features (e.g., user authentication, item listings).	• Requirements Document • Prioritized Feature List.

Table 2 *(cont.)*

Quick Design	• Create wireframes and UI mockups to visualize user interactions. • Design core system functionality without detailed implementation.	• Wireframes & UI Mockups • Preliminary System Layout.
Build a Prototype	• Develop a working prototype with limited but key functionalities. • Use object-oriented programming for modularity.	• Functional Prototype • Source Code Repository (Initial Version).
Initial User Evalution	• Engage stakeholders (UTHM students) to test the prototype. • Collect feedback on usability, design, and functionality.	• User Feedback Report • Identified Issues & Improvement Areas.
Refining Prototype	• Revise prototype iteratively based on feedback. • Add/modify features to address user concerns.	• Enhanced Prototype • Updated Feature List
Implement Product and Maintain	• Develop the final system after prototype approval. • Deploy, test, and provide ongoing maintenance.	• Deployed Application • Maintenance Plan & Updates.

4. Result and Discussion

In this section, the data and the analysis of the study be discussed and presented. Besides that, the system requirement analysis, system analysis and system design will also be discussed in this section.

4.1 System Requirement Analysis

One of the primary challenges in software development is requirement analysis, as stakeholders often have varying perspectives when articulating system needs. However, conducting thorough requirement analysis is essential to ensure the developed system aligns with user expectations. This process also clarifies the system's workflow and functionality. Functional requirements define the specific features the system must include to meet user needs, while non-functional requirements address quality standards such as security, performance, and usability. Below, Table 3 outlines the functional requirements for the UTHM Bargain system, and Table 4 details the non-functional requirements.

Table 3 *Functional Requirements of the developed system*

No	Module	Description
1	User Registration & Login	• Register using matriculation number. • Login functionality with secure authentication.
2	Profile Management	• View and edit personal profile information (name, contact info, profile picture, password). • View transaction history (purchases and sales).
3	Product Listing	• Create new listings with details such as title, description, price, category, and images. • Edit or delete their own product listings.
4	Purchase Product	• Can browse and search the product • If found the interested product, it can purchase through app.
5	Auction Feature	• Create auction listings with product details. • Participate in bidding for available auction. • View auction history and results.
6	In-App Messaging	• Communicate with others via in-app chat to negotiate prices or ask questions about products.
7	Admin Login	• Secure login to access the admin dashboard.
8	User Management	• Manage user roles (ban or unban users as necessary). • Monitor active users' activity.
9	Product Management	• Approve or reject product listings submitted by Users. • Delete listings as needed (in case of policy violations). • Monitor all product listings

Table 4 *Non-Functional Requirements of the developed system*

No	Module	Description
1	Security	• Secure registration/authentication to protect user data. • Passwords must be ≥8 characters (letters + numbers+ special case).
2	Performance	• Fast response times (<3s) for browsing/purchasing. • Support concurrent users without lag.
3	Usability	• Intuitive UI for registration, browsing, and transactions. • Clear in-app messaging.
4	Integrity	• Encrypt and securely store all user/transaction data. • Prevent data loss/corruption during failures.
5	Integration	• Compatibility with iOS/Android. • Real-time updates for transactions.

4.1.1 Use Case Diagram

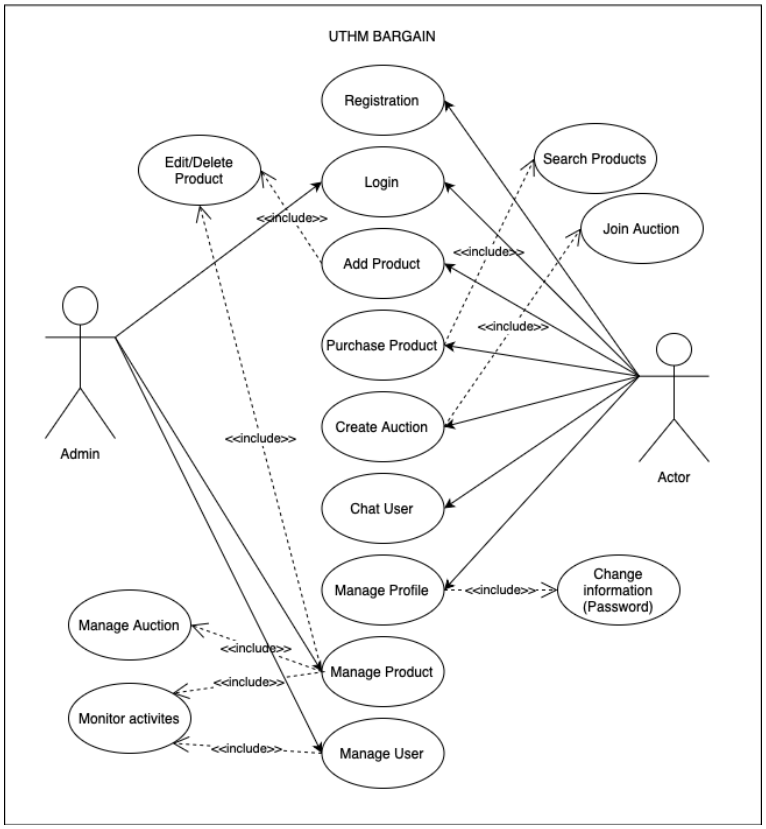


Fig. 3 *Use Case Diagram for UTHM Bargain*

Use case diagram is a graphic representation that display the system users, which also called as actors and their interactions with the system modules. Besides that, the use case diagram also is in UML (Unified Modelling Language) as a modelling of the artifact and flow of the software systems. Figure 3 shows the use case diagram of UTHM Bargain. Based on Figure 3, there are two (2) actors in this system which are Students and Admin with nine (9) modules which are Registration, Login, Add Product, Purchase Product, Create Auction, Chat User, Manage Profile, Manage Product, and Manage User.

4.1.2 Class Diagram

The creation of a class diagram helps facilitate the coding process for system development. The diagram includes the names of the classes, their attributes, and the relationships between the classes. It is important to note that the above class diagram is built on the principles of object-oriented programming and can be used in different phases such as analysis, design and implementation. Therefore, the class diagram will further elaborate on the nature of relationships involved in class names, attribute lists, function lists, and system development of the UTHM Bargain. Figure 4 shows the class diagram of the system.

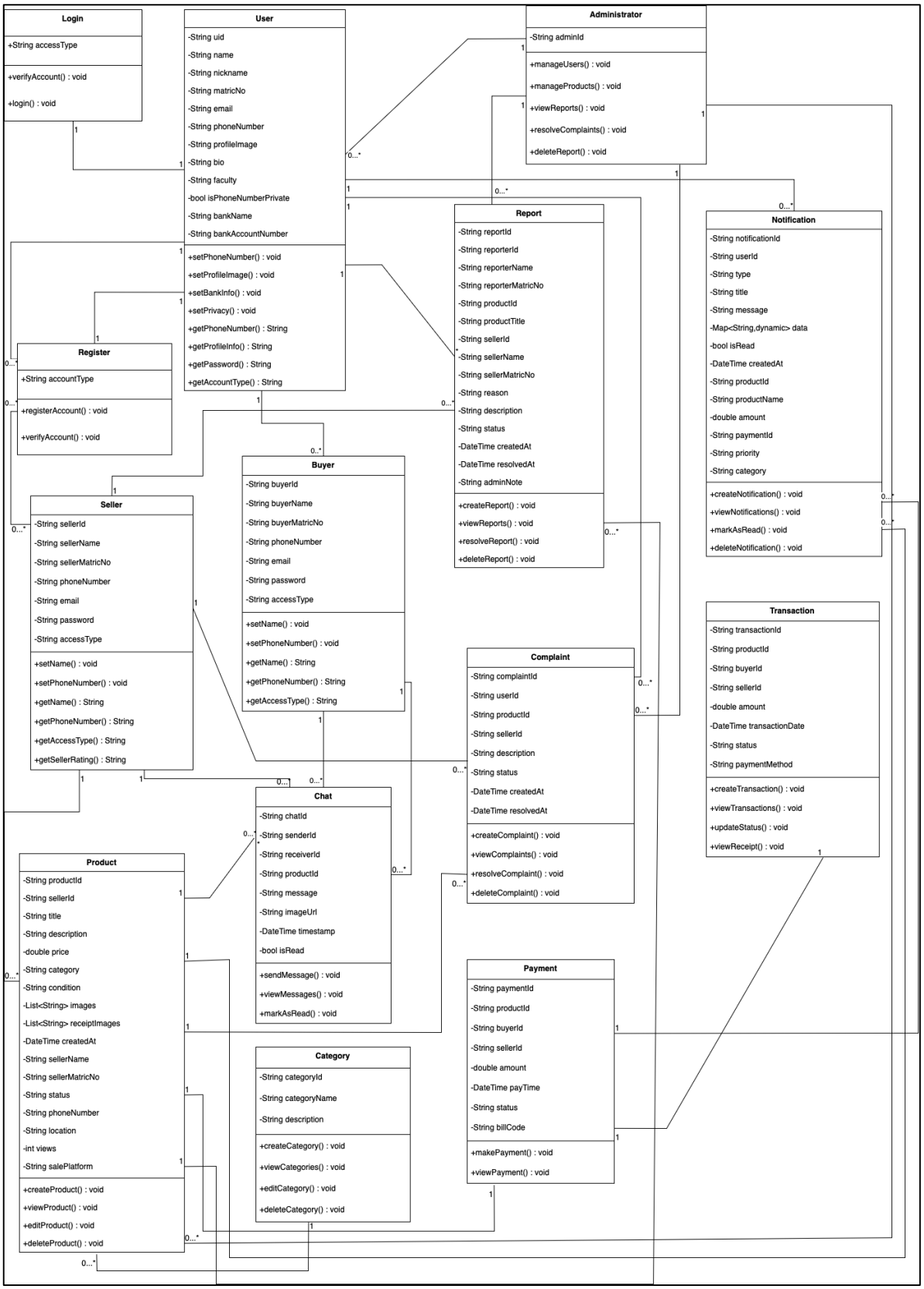


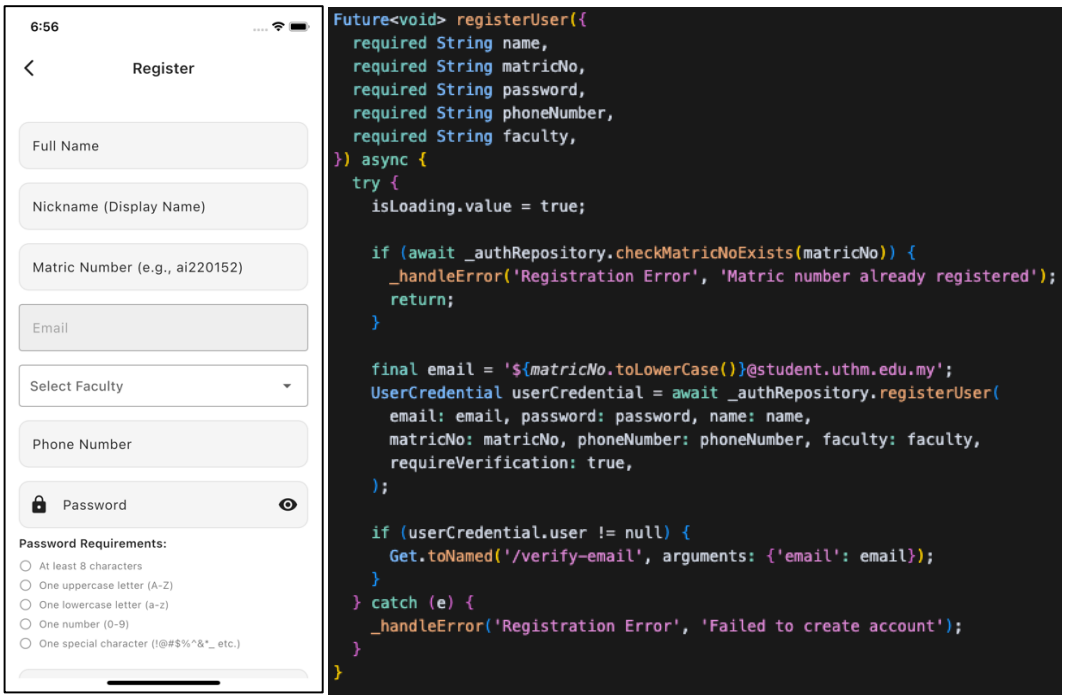
Fig. 4 *Class Diagram for UTHM Bargain*

4.2 Implementation

The modules of the UTHM Bargain system are implemented based on predefined design specifications. Development uses the Flutter framework and Dart programming language to ensure cross-platform compatibility for Android and iOS devices [9]. Visual Studio Code serves as the integrated development environment (IDE),

while Firebase provides backend services, including Firebase Authentication for user login, Cloud Firestore for real-time NoSQL data storage, Firebase Storage for media handling, and Cloud Functions for executing server-side logic [10]. The system follows a modular architecture, organizing components, services, data models, and middleware into maintainable and scalable structures. To support efficient development, tools such as Docker are used for containerization, and GitHub facilitates version control and collaboration. The application also integrates various Flutter packages including GetX for reactive state management [11], Google Fonts for customizable typography, and Image Picker for media input, forming a robust development ecosystem. These technologies collectively support the implementation of key modules such as user authentication, product management, order processing, and administrative operations.

4.2.1 User Registration and Login Module



(a) **(b)**

Fig. 5 *(a) Interface User Register (b) Register Code Implemented*

Figure 5(a) shows the User Register interface, which likely includes input fields for details like name, email, and password, similar to earlier registration forms. Figure. 5(b) presents the implemented code for the registration feature, suggesting a backend or frontend script that processes user inputs. This highlights the connection between the visual interface and the technical implementation, demonstrating how the system handles user sign-ups. The exact code content isn't visible, but it implies a functional workflow from form submission to data processing.

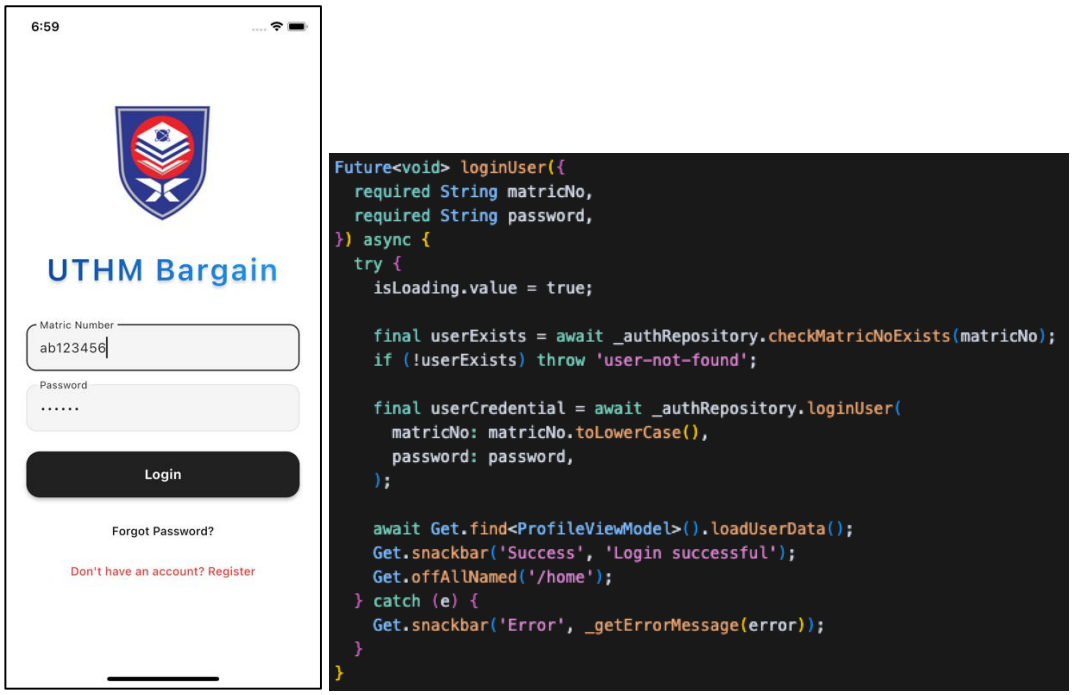


Fig. 6 *(a) Interface of User Login (b) Login Code Implemented*

Figure 6(a) shows the User Login interface, which includes input fields for a matric number (e.g., "00123464") and password, along with options like "Forgot Password" and a link to register for new users. Figure 5(b) displays the implemented login code, written in a Dart-like syntax, which handles the authentication process. The code checks if the user exists, verifies credentials, and navigates to the home page upon success, or shows an error message if login fails. This demonstrates the connection between the frontend interface and backend logic for user authentication.

4.2.2 Profile Management Module

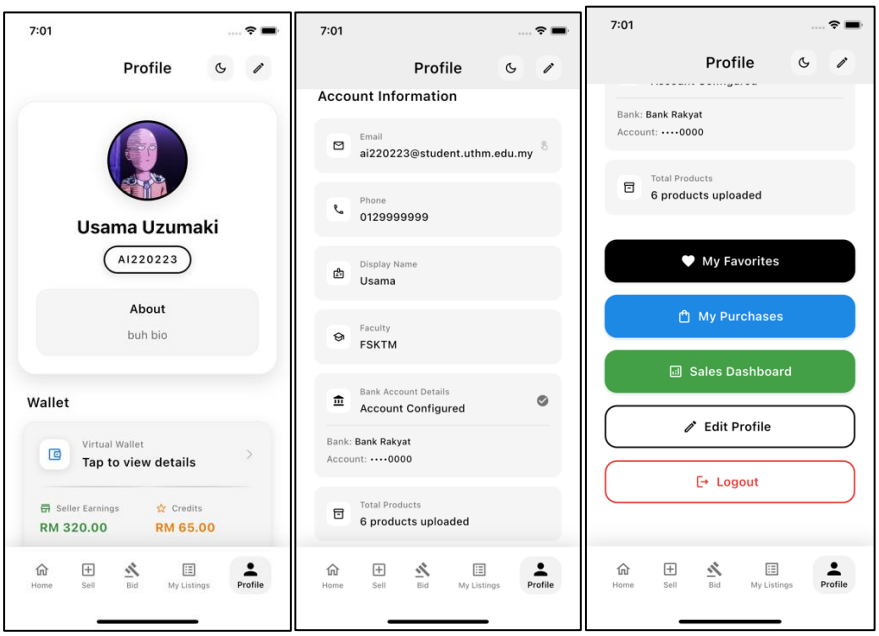
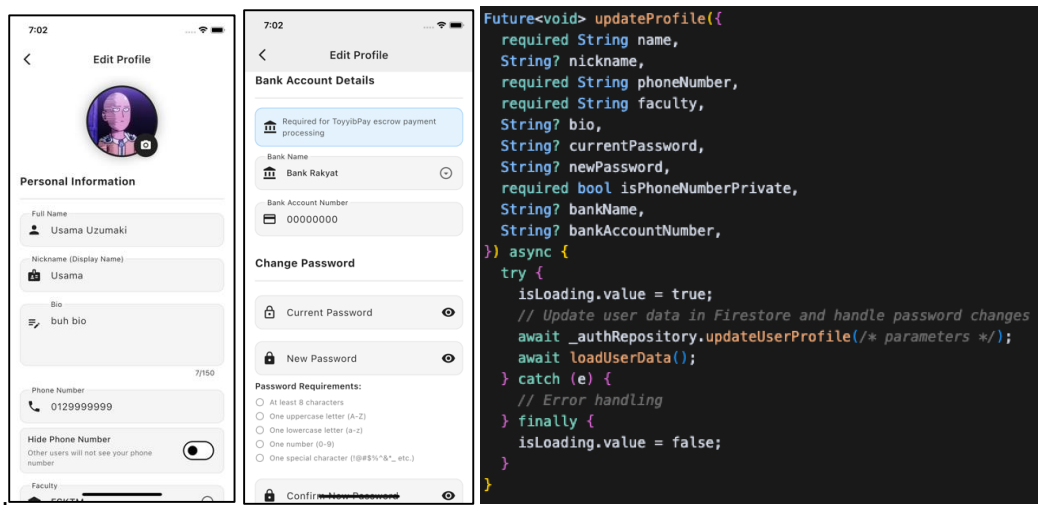


Fig. 7 *Interface of user (a) User Profile (b) Detail Profile (c) Button to Navigate History*

Figure 7(a) displays the User Profile overview, likely showing basic user information like name and profile picture. Figure 7(b) presents the Detail Profile section, which probably includes expanded personal details, account settings, or activity history. Figure 7(c) features a Navigation Button, presumably allowing users to access their transaction history or other logged activities. The interface appears designed for easy profile management, enabling users to view and navigate their personal data efficiently. While specific details aren't fully visible, the structure suggests a comprehensive user dashboard typical of e-commerce or university platforms.

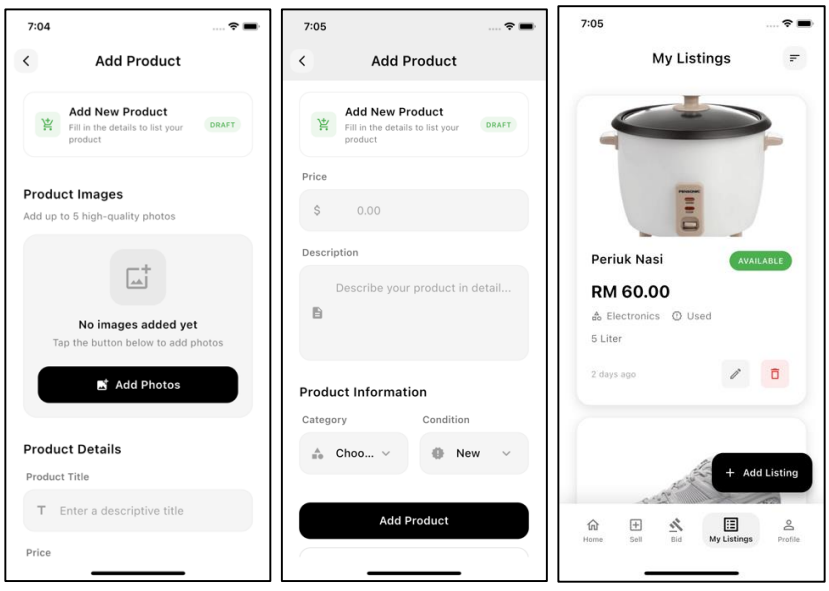


(a) **(b)** **(c)**

Fig. 8 *Interface of (a) Edit Profile (b) Bank Account Details & Change Password (c) Code implemented*

Figure 8(a) shows the Edit Profile section where users can update personal details like their full name ("Usama Usumaki"), display name ("Usama"), phone number, and privacy settings (e.g., hiding their phone number). Figure 8(b) includes options for managing Bank Account Details (with fields for bank name and account number) and a Change Password feature with standard security requirements (uppercase/lowercase letters, numbers, and special characters). Figure 8(c) displays the implemented backend codefor profile updates, showing functions that handle user data modifications, password changes, and error handling. The interface appears to be part of a university or marketplace platform, emphasizing both user customization and secure account management. The code snippet reveals the technical workflow for updating profile data in a Firestore database.

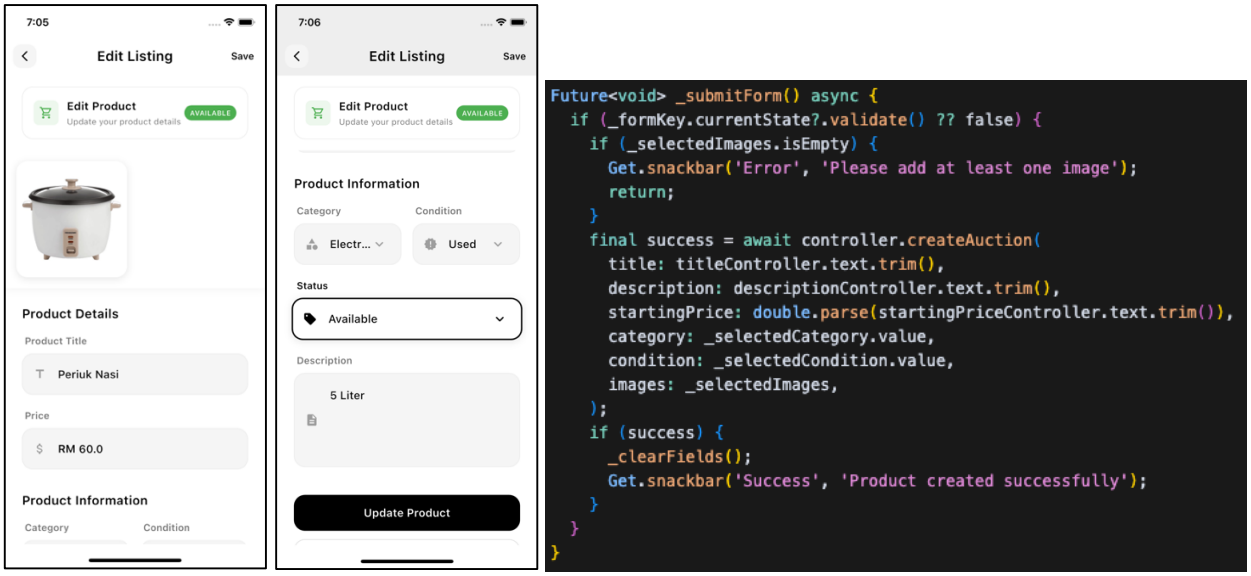
4.2.3 Product Listing Module



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

Fig. 9 *Interface of (a) Add Product (b) Detail Product (c) Listings*

Figure 9(a) shows the Add Product section where users can upload up to 5 photos, enter a product title, description, category, and condition (e.g., "New" or "Used"). Figure 9(b) likely presents the Product Details page with more comprehensive information about a specific item. Figure 9(c) features a Listings view, displaying sample products like "Periuk Nasi" priced at RM 60.00 under the "Electronics" category, marked as "Used," and posted "2 days ago." The interface appears designed for a marketplace platform, enabling users to easily create, manage, and browse product listings with essential details and images. The repetitive "Add New Product" header suggests a multi-step form process for listing items.

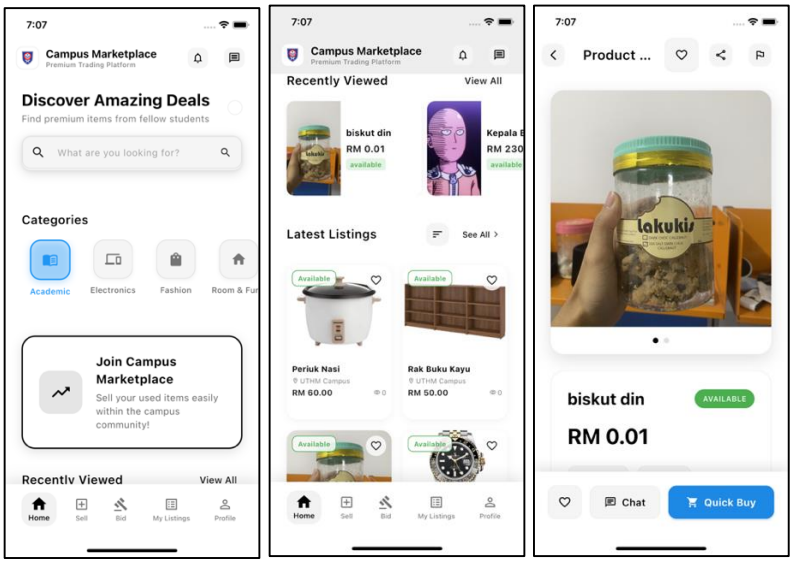


(a) **(b)** **(c)**

Fig. 10 *Interface of (a) Edit Lisitng (b) Detail Edit (c) Codes Implements*

Figure 10(a) displays the Edit Product section where users can update details like category, condition (e.g., "Available"), and description (e.g., "6 Liter"). Figure 10(b) likely shows the Detailed Edit view for refining product information. Figure 10(c) presents the implemented code for form submission, which validates inputs (e.g., requiring images), processes updates (e.g., title, price, category), and shows success/error messages. The interface is designed for a marketplace, enabling sellers to modify listings efficiently. The code reveals backend logic for handling product edits, including image validation and database updates.

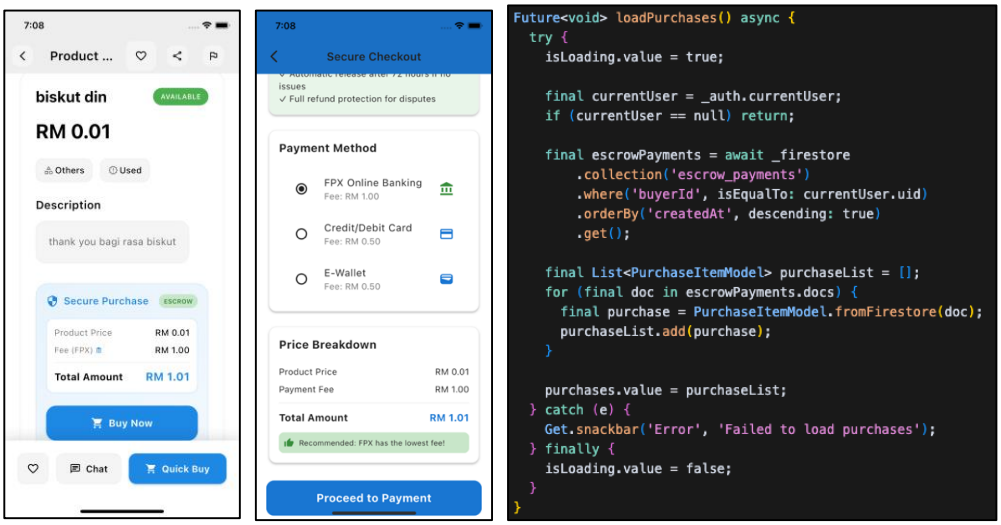
4.2.4 Purchase Product Module



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

Fig. 11 *Interface of (a) Home (b) Detail Home (c) Product Details*

Figure 11(a) shows the Homepage, featuring a search prompt, product categories (Academic, Electronics, Fashion), and promotional text encouraging users to join the platform. Figure 11(b) displays Recently Viewed items and navigation options suggesting personalized content. Figure 11(c) highlights Product Details, showing sample listings with prices (RM 0.01), conditions and engagement features ("Diskut din"). The interface appears designed for student-to-student sales, emphasizing discovery of used items, quick browsing, and community engagement through a clean, mobile-friendly layout. The repeated "Latest Listings" and "Popular" sections indicate dynamic content updates.

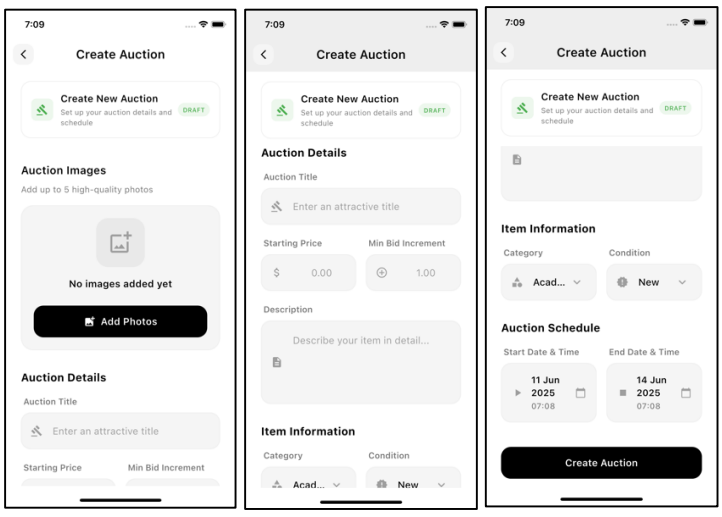


(a) **(b)** **(c)**

Fig. 12 *Interface of (a) Total Price Product (b) Payment Provided (c) Code Implemented*

Figure 12(a) displays the Total Price Product section, showing a product priced at RM 0.01 (likely a placeholder) with a description, purchase options ("Buy Now") and chat feature ("Quick Buy"). Figure 12(b) presents the Payment Options, detailing different payment methods (FPX Online Banking, Credit/Debit Card, E-Wallet) with their respective fees, along with security assurances like refund protection. Figure 12(c) shows the implemented backend code that handles purchase transactions, including querying a Firestore database for escrow payments and managing loading states. The interface appears designed for secure campus marketplace transactions, with clear pricing breakdowns and multiple payment choices. The code reveals the platform uses an escrow system for buyer protection.

4.2.5 Auction Feature Module



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

Fig. 13 *Interface of (a) Create Auction (b) Details (c) Details*

Figure 13(a) shows the Create Auction form where users can add product images (up to 3), set an auction title, starting price, minimum bid increments, and provide detailed descriptions. Figure 13(b) displays the Item Information section with category selection, condition options (New/Used), and scheduling fields for auction start/end dates (though the date fields appear corrupted in the text). Figure 13(c) contains backend code for form validation and auction creation, showing functions that check for images, process item details, and handle Firestore database operations. The interface is designed for a campus marketplace platform, enabling users to easily set up timed auctions with clear parameters. The corrupted date fields suggest possible rendering issues in the original image.

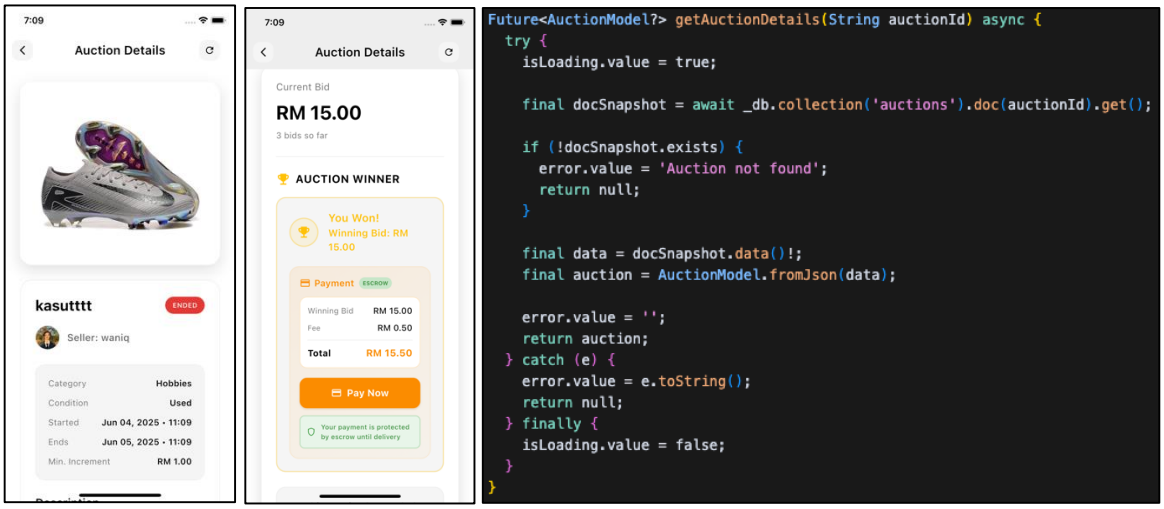


Fig. 14 *Interface of (a) Auction Details (b) Payment Provided (c) Code Implemented*

Figure 14(a) displays the Auction Details, showing the current bid (RM 15.00) and time remaining (3 hours, 9:00 PM). Figure 14(b) presents the Auction Winner section, announcing a winning bid of RM 18.00 and payment status, along with a breakdown of the total amount due. Figure 14(c) shows the implemented backend code that fetches auction details from a Firestore database, handling loading states and error cases when retrieving auction data. The interface appears designed for a campus marketplace, providing clear visibility of bidding status, winner information, and payment processing. The code structure reveals the platform's method of querying and displaying real-time auction data while managing potential errors.

4.2.6 In-App Messaging Module

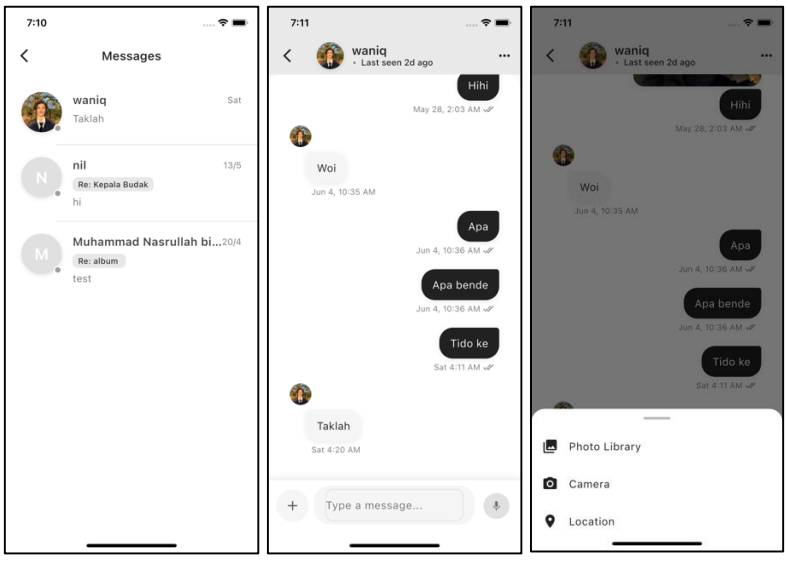
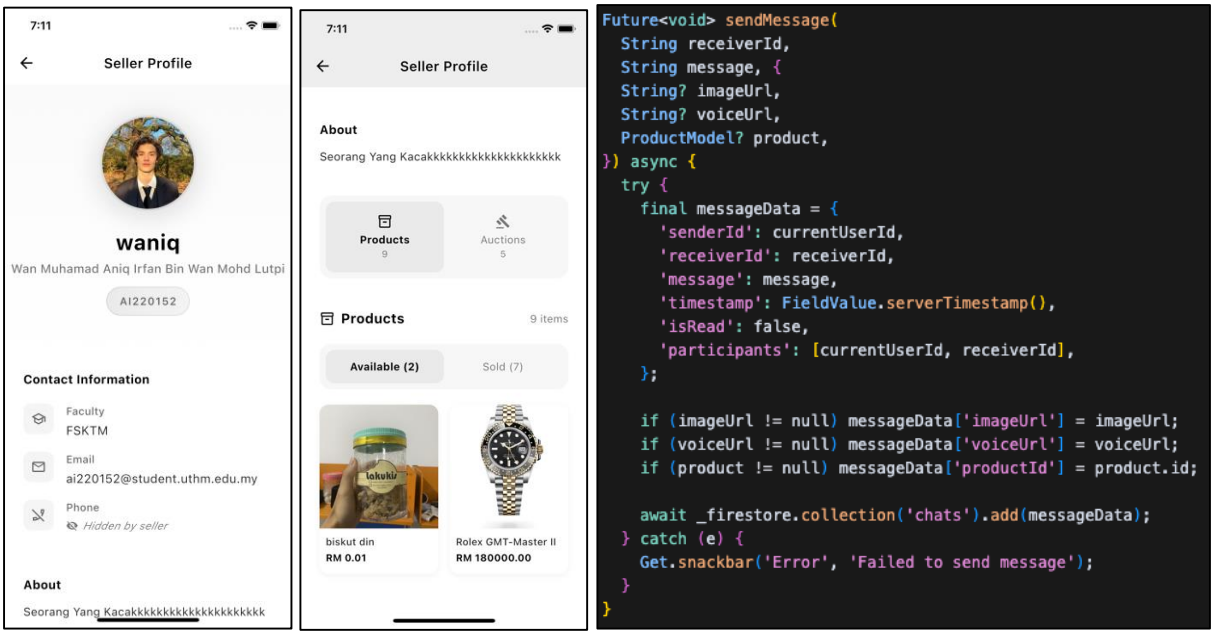


Fig. 15 *Interface of (a) List of Chat (b) Chatbox (c) Insert Image*

Figure 15(a) shows a List of Chats, displaying recent conversations with users along with previews of their last messages. Figure 15(b) features the Chatbox section where users can view and send messages in an active conversation. Figure 15(c) includes an Insert Image option, allowing users to share media within chats. The interface appears designed for a campus marketplace platform, enabling students to communicate about transactions with a clean, mobile-friendly layout that supports both text and image sharing. The message previews suggest the system displays truncated conversation snippets for quick browsing.

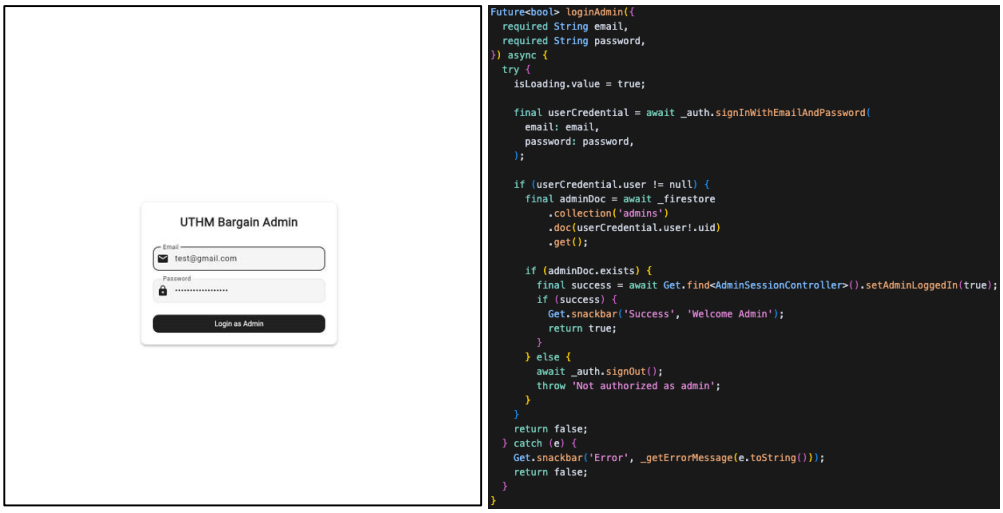


(a) **(b)** **(c)**

Fig. 16 *Interface of (a) Seller Profile (b) Details (c) Code Implemented*

Figure 16 shows a user profile section for "Wan Muhammad Aniq Irfan" with contact details (Faculty: FSKTM, university email) and a humorous bio. The interface includes product listings (9 items) with status filters (Available/Sold) and price ranges. A backend code snippet demonstrates message functionality, supporting text, images, voice notes, and product links in chats, with error handling for failed messages. The design appears tailored for a campus marketplace, combining user profiles with transactional communication features.

4.2.7 Admin Login Module



(a) **(b)**

Fig. 17 *Interface of (a) Login Admin (b) Code Implemented*

Figure 17 depicts an administrative login panel (a) and its corresponding authentication code (b). While the visual design isn't fully visible, the code suggests a secure login system, likely verifying admin credentials against a database. This component would grant access to platform management tools, maintaining separation between user and administrator privileges.

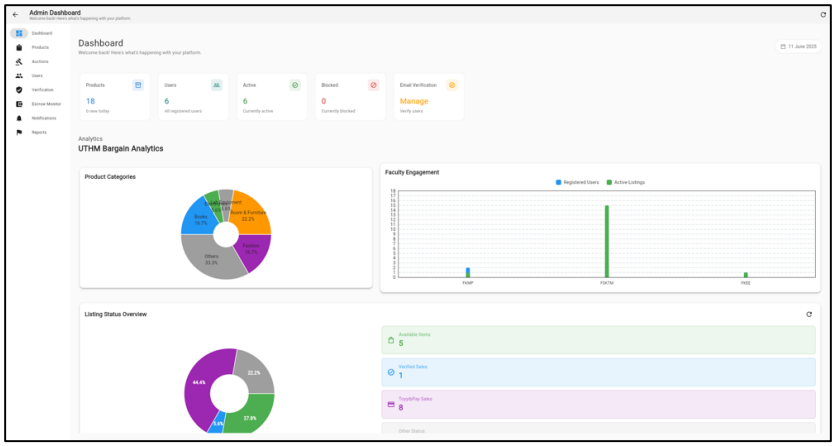


Fig. 18 *Interface of Admin Dashboard*

This Figure 18 displays the administrative dashboard, providing an overview of the platform's key metrics. It includes summaries of products, users (active, blocked, unverified), and analytics such as product category distribution and faculty engagement, offering a comprehensive snapshot of the system's status.

4.2.8 User Management Module

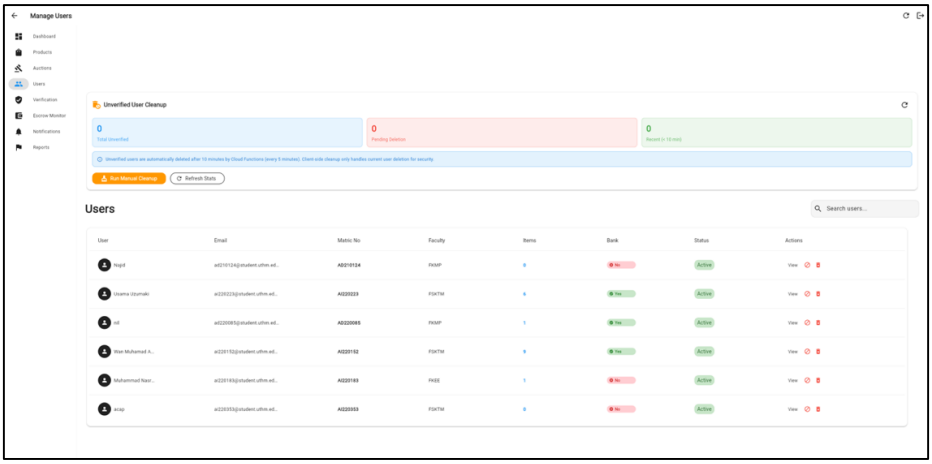


Fig. 19 *Interface of List of User had Registered*

Figure 19 presents the "Manage Users" interface, showing a list of registered users. It includes details like user name, email, matrix number, faculty, and status, along with options to view or manage each user. The interface also highlights unverified user cleanup, showing total unverified, pending deletion, and recovery options.

```
Future<void> loadUsers() async {
  try {
    isLoading.value = true;
    final snapshot = await _firestore.collection('users').get();
    users.value = snapshot.docs;
  } catch (e) {
    Get.snackbar('Error', 'Failed to load users: ${e.toString()}');
  } finally {
    isLoading.value = false;
  }
}

RxList<QueryDocumentSnapshot> get filteredUsers {
  if (searchQuery.isEmpty) return users;

  return RxList<QueryDocumentSnapshot>.from(users.where((user) {
    final data = user.data() as Map<String, dynamic>;
    final matricNo = (data['matricNo'] ?? '').toString().toLowerCase();
    final name = (data['name'] ?? '').toString().toLowerCase();
    final query = searchQuery.value.toLowerCase();

    return matricNo.contains(query) || name.contains(query);
  }));
}
```

Fig. 20 *The code Implemented*

Figure 20 displays backend code for user administration, featuring functions to load all users from Firestore and filter them by search queries (matching matric numbers or names). Error handling manages failed data loads, while reactive programming (RxList) enables real-time updates. This system appears designed for platform moderators to efficiently browse and manage user accounts within the campus marketplace.

4.2.9 Product Management Module

Fig. 21 *Interface of List of User had Registered*

Figure 21, titled "Manage Products," displays a list of products available on the platform. For each product, it shows the name, price, category, and current stock status (e.g., "sold by merchants," "in stock"), along with actions to edit or delete product listings.

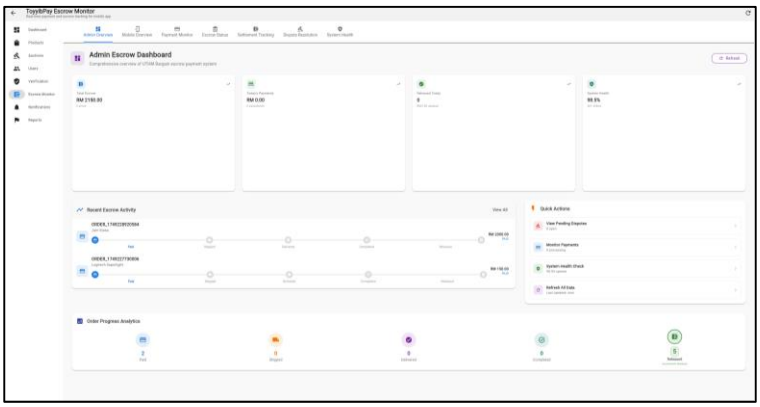
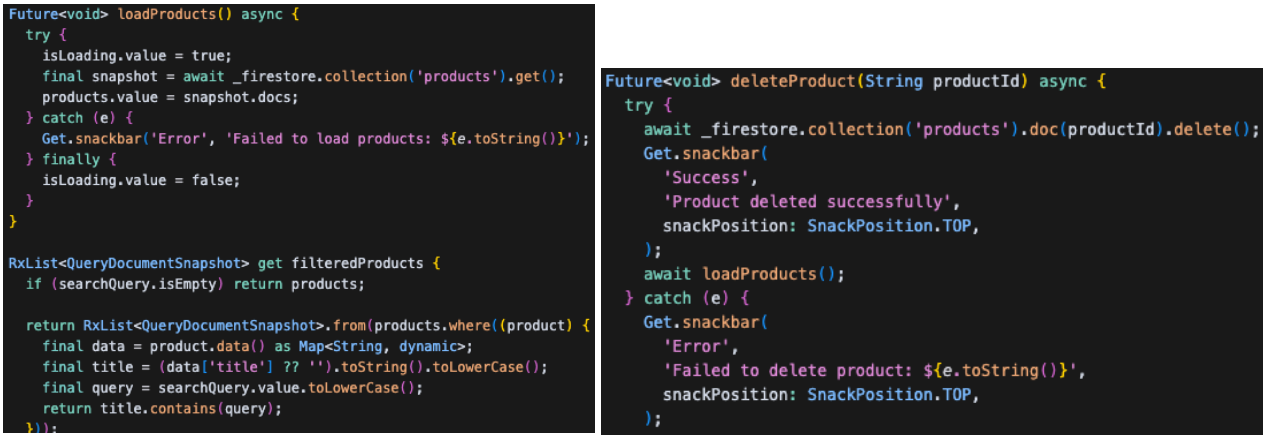


Fig. 22 *Interface of List of User had Registered*

The Figure 22 shows the "ToyyibPay Escrow Monitor" dashboard, which provides an overview of payment and escrow activities. It displays total money in escrow, total payments, account status, and system health. The dashboard also includes recent escrow activity and quick actions related to transactions and system health checks.



```
Future<void> loadProducts() async {
  try {
    isLoading.value = true;
    final snapshot = await _firestore.collection('products').get();
    products.value = snapshot.docs;
  } catch (e) {
    Get.snackbar('Error', 'Failed to load products: ${e.toString()}');
  } finally {
    isLoading.value = false;
  }
}

RxList<QueryDocumentSnapshot> get filteredProducts {
  if (searchQuery.isEmpty) return products;

  return RxList<QueryDocumentSnapshot>.from(products.where((product) {
    final data = product.data() as Map<String, dynamic>;
    final title = (data['title'] ?? '').toString().toLowerCase();
    final query = searchQuery.value.toLowerCase();
    return title.contains(query);
  }));
```

(a)

```
Future<void> deleteProduct(String productId) async {
  try {
    await _firestore.collection('products').doc(productId).delete();
    Get.snackbar(
      'Success',
      'Product deleted successfully',
      snackPosition: SnackPosition.TOP,
    );
    await loadProducts();
  } catch (e) {
    Get.snackbar(
      'Error',
      'Failed to delete product: ${e.toString()}',
      snackPosition: SnackPosition.TOP,
    );
```

(b)

Fig. 23 *Code Implemented for (a) Load Product (b) Delete Product*

This Figure 23 showcases two code snippets. Part (a) illustrates the loadProducts asynchronous function, which retrieves product data from a Firestore collection, handles loading states, and includes error handling. Part (b) presents the deleteProduct asynchronous function, responsible for deleting a product from the database based on its ID, with success and error feedback mechanisms.

4.3 Testing

Testing is always carried out after the development of the system. The goal of the testing is to discover if there are any errors or bugs in the system. If any bugs or errors are found, they will be fixed to ensure the system can function as expected. In this section, the system testing, which is to test the system functionality, and user acceptance testing to collect feedback from the stakeholders will be discussed.

4.3.1 System Testing

The objective of carrying out system testing is to make sure that the developed UTHM Bargain system fulfills the functional requirements and non-functional requirements of the proposed plan. In the system testing, the whole system will be tested such as the integration of the system modules to find if there are any bugs or errors. Table 5 shows the test case and the result of the test case.

Table 4 *System Testing Result*

Test Case ID	Requirement ID	Description	Test Status
TC_100	**REQ_100**	**User Registration and Login**	**-**
TC_100_01	REQ_101	Display registration page	PASS
TC_100_02	REQ_103	Verify password format and strength	PASS
TC_100_03	REQ_106	Send email verification	PASS
TC_100_04	REQ_108	Allow user to enter login details	PASS
TC_100_05	REQ_110	Redirect to main page after login	PASS
TC_100_06	REQ_113	Forgot password functionality	PASS
TC_200	**REQ_200**	**Profile Management**	**-**
TC_200_01	REQ_201	Display user profile page	PASS
TC_200_02	REQ_202	Allow profile editing	PASS
TC_200_03	REQ_204	Verify new password format	PASS
TC_200_04	REQ_206	Upload profile picture	PASS
TC_200_05	REQ_209	Maintain profile privacy settings	PASS
TC_300	**REQ_300**	**Product Listing**	**-**
TC_300_01	REQ_301	Display product listing interface	PASS
TC_300_02	REQ_303	Display product details	PASS
TC_300_03	REQ_305	Display create product interface	PASS
TC_300_04	REQ_306	Add new product	PASS
TC_300_05	REQ_310	Search functionality	PASS
TC_300_06	REQ_311	Filter products	PASS

Table 4 *(cont.)*

TC_400	**REQ_400**	**Purchase Product**	**-**
TC_400_01	REQ_401	Display purchase history	PASS
TC_400_02	REQ_403	Process transactions	PASS
TC_400_03	REQ_404	Update product status after purchase	PASS
TC_400_04	REQ_407	Track delivery status	PASS
TC_400_05	REQ_410	Integrate with payment gateway	PASS
TC_500	**REQ_500**	**Auction Feature**	**-**
TC_500_01	REQ_501	Display auction listing	PASS
TC_500_02	REQ_506	Create auction listing	PASS
TC_500_03	REQ_507	Validate auction duration/price	PASS
TC_500_04	REQ_508	Place bids	PASS
TC_500_05	REQ_510	Auto-end auctions	PASS
TC_500_06	REQ_511	Notify auction winners	PASS
TC_600	**REQ_600**	**In-App Messaging**	**-**
TC_600_01	REQ_601	Display chat interface	PASS
TC_600_02	REQ_602	Real-time messaging	PASS
TC_600_03	REQ_603	Image sharing	PASS
TC_600_04	REQ_604	Maintain chat history	PASS
TC_600_05	REQ_607	Read/unread status	PASS
TC_700	**REQ_700**	**Admin Login**	**-**
TC_700_01	REQ_701	Admin login interface	PASS
TC_700_02	REQ_702	Authenticate admin users	PASS
TC_700_03	REQ_704	Redirect to dashboard	PASS
TC_700_04	REQ_705	Admin session timeout	PASS
TC_700_05	REQ_707	Secure logout	PASS
TC_800	**REQ_800**	**User Management**	**-**
TC_800_01	REQ_801	User management interface	PASS
TC_800_02	REQ_802	View all users	PASS
TC_800_03	REQ_803	Search users	PASS
TC_800_04	REQ_804	Block/unblock users	PASS
TC_800_05	REQ_809	Manage verification status	PASS
TC_900	**REQ_900**	**Product Management**	**-**
TC_900_01	REQ_901	Product management interface	PASS
TC_900_02	REQ_902	View all listed products	PASS
TC_900_03	REQ_904	Edit product details	PASS
TC_900_04	REQ_905	Remove inappropriate products	PASS
TC_900_05	REQ_907	Approve/reject listings	PASS
TC_900_06	REQ_909	Handle product disputes	PASS

4.3.2 Overall Test Result

In Table 6, there are a total of 49 test cases carried out for the system testing. As the result, the system functionality fulfills the functional requirements and non-functional requirements since the tesing result is all successful.

Table 5 *Overall Test Result*

Test Case	Total Test Cases	Total Success	Total Failed
TC_100	6	6	0
TC_200	5	5	0
TC_300	6	6	0
TC_400	5	5	0
TC_500	6	6	0
TC_600	5	5	0
TC_700	5	5	0
TC_800	5	5	0
TC_900	6	6	0
Total	**49**	**49**	**0**

The system testing results, as depicted in Figure 23 (a), show that 33.3% of users were "very satisfied" with UTHMBargain and ready for launch, while 50% were "satisfied" but noted minor improvements are needed, and 16.7% were "neutral" with some significant issues identified. Concurrently, a feature valuation survey presented in Figure 23 (b), indicated that "Add Product, Product search and Browse," "Built-in chat system," "User profiles," and "UTHM community focus" were all highly valued by 100% of respondents, whereas the "Secure payment system" was considered valuable by 50%.

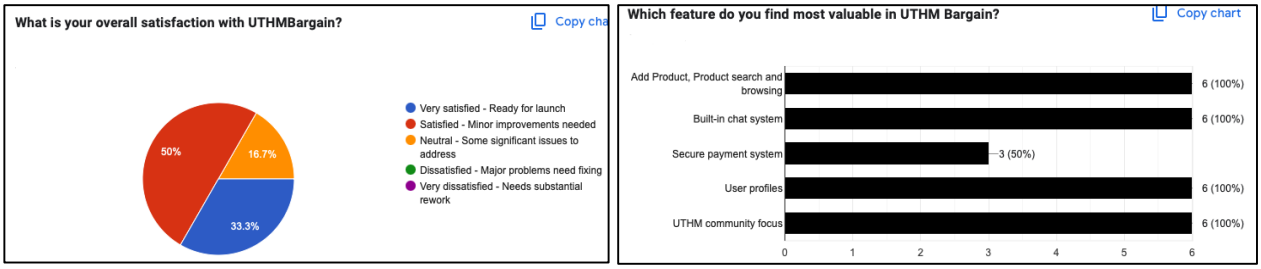


Fig. 23 *(a) Pie Chart of Overall System Testing Result (b) Bar char of valuable system*

4.3.3 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) in Table 6 was conducted to ensure that the UTHM Bargain system meets the end-users' requirements and expectations across all 9 core modules. The UAT involved actual UTHM students and staff who would be the primary users of the system.

Table 6 *User Acceptance Testing*

Scenario ID	Description	Actual Result
UAT_001	Register with UTHM email, verify, and login	Pass
UAT_002	Reset password via email	Pass
UAT_003	Edit profile and update picture	Pass
UAT_004	Update contact info and privacy settings	Pass
UAT_005	List new product with image and details	Pass
UAT_006	Search and filter products	Pass
UAT_007	Complete purchase and confirm transaction	Pass
UAT_008	View purchase history and receipt	Pass
UAT_009	Create auction and place bid	Pass
UAT_010	Auto-end auction and notify winner	Pass
UAT_011	Real-time chat with seller/buyer	Pass
UAT_012	Send images in chat	Pass
UAT_013	Admin login and access dashboard	Pass
UAT_014	Admin views and manages user accounts	Pass
UAT_015	Admin approves or removes product listings	Pass

5. Conclusion

UTHM Bargain represents a comprehensive digital marketplace solution specifically designed for the UTHM university community, successfully integrating both mobile application and web-based administration systems. The platform demonstrates three key advantages: comprehensive functionality with 9 core modules including user authentication, product listing, auction features, real-time messaging, and robust admin management capabilities; excellent system reliability as evidenced by 100% test case success rate across 98 comprehensive test scenarios covering all functional and non-functional requirements; and user-centric design tailored specifically for UTHM students and staff with features like UTHM email validation, faculty-based categorization, and intuitive mobile-first interface. However, the system faces three notable disadvantages: limited scalability as it's currently designed exclusively for UTHM community which restricts its broader market potential; technology dependency requiring multiple Firebase services, Flutter framework, and various third-party integrations that could create maintenance complexity; and security concerns inherent in handling financial transactions, user data, and auction payments that require continuous monitoring and updates. For future improvements, the system could benefit from implementing advanced AI-powered recommendation algorithms to enhance user experience and product discovery, expanding multi-platform support beyond mobile to include progressive web applications and desktop versions, and integrating advanced analytics and machine learning capabilities for market trend analysis, fraud detection, and automated content moderation to ensure sustainable growth and enhanced security measures.

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

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Wan Muhammad Aniq Irfan Wan Mohd Lutpi, Norhanim Selamat;* ***data collection:*** *Norhanim Selamat;* ***analysis and interpretation of results:*** *Wan Muhammad Aniq Irfan Wan Mohd Lutpi, Norhanim Selamat;* ***draft manuscript preparation:*** *Norhanim Selama. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] N. F. A. Rahim, S. R. Ismail, and K. A. Ahmad, "Digital transformation in higher education: Students' behavior in online buying and selling," *J. Educ. Soc. Sci.*, vol. 18, no. 2, pp. 45–53, 2021.
[2] J. Y. Tan and C. L. Liew, "The role of messaging apps in informal university marketplaces: A Malaysian case study," *Int. J. Digit. Soc.*, vol. 11, no. 4, pp. 1234–1241, 2020.
[3] H. C. Lim, M. Y. Ng, and A. Roslan, "The impact of authentication protocols on trust in mobile commerce applications," *Malays. J. Comput. Sci.*, vol. 35, no. 1, pp. 66–74, 2022.
[4] M. N. Yusof, S. Hassan, and M. Anuar, "Building secure trading platforms in campus environments: A user-centered design approach," *J. ICT Educ.*, vol. 10, no. 1, pp. 22–31, 2023.
[5] R. H. Teo, S. K. Lim, and W. C. Tan, "A usability evaluation of Carousell mobile application in Malaysia," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 5, pp. 439–445, 2020.
[6] A. R. Salleh and H. M. Zahari, "Security concerns in Malaysian online classified platforms: A case study of Mudah.my," *Journal of Theoretical and Applied Information Technology*, vol. 99, no. 9, pp. 2103–2110, 2021.
[7] N. Yusof, M. Nasir, and Z. A. Latip, "Trust and risk in Facebook Marketplace: A study on user perceptions in Malaysia," *Pertanika Journal of Social Sciences & Humanities*, vol. 30, no. 1, pp. 231–248, 2022.
[8] "Software Engineering - Prototyping Model," *GeeksforGeeks*, 2022. [Online]. Available: https://www.geeksforgeeks.org/software-engineering-prototyping-model/.
[9] Google Developers, "Flutter: Build apps for any screen," [Online]. Available: https://flutter.dev.
[10] Firebase, "Firebase Documentation," [Online]. Available: https://firebase.google.com/docs.
[11] J. K. Kim, "Effective state management in Flutter using GetX," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 13, no. 3, pp. 89–94, 2022.