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

AITCS

Vol. 6 No. 2 (2025) 528-547
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

A Medication Reminders and Disposal Mobile Application Using Data Encryption Technique

Clement Foong Wen Kai[1], Sofia Najwa Ramli[1]*

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

*Corresponding Author: sofianajwa@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.029

Article Info

Received: 1 August 2025
Accepted: 19 November 2025
Available online: 30 November 2025

Keywords

Medication, Flutter, Large Language Model, Artificial Intelligence, Google Gemini, Google Firebase, Grounding, Agents

Abstract

Medication non-adherence and improper disposal pose significant challenges to society and environment particularly in Malaysia. Therefore, mobile application MediCare+ has been introduced to address these issues by combining medication reminders, disposal guidance, and an Artificial Intelligence (AI) chatbot into a single, user-friendly platform. While current applications primarily focus on medication reminders, MediCare+ provides a comprehensive solution with personalized assistance and proper disposal information. Developed using Flutter and Firebase, the application's core innovation is an AI agent powered by the Gemini API with function calling. This allows it to perform advanced tasks like searching trusted medical sources and locating nearby disposal sites. MediCare+ also provides AES-256-CBC encryption for sensitive medicine data. A prototyping methodology was used to develop the application. MediCare+ was developed primarily emphasis on Android. The development of MediCare+ will assist patients with adherence to their prescriptions and greatly reduce the pollution caused by improper disposal of medicines.

1. Introduction

Medication adherence refers to the degree to which patients take their medications as prescribed by healthcare professionals [1]. However, there are many individuals who do not follow their prescriptions and end up with unused medicine which often expires over time. There was a study that out of the 1184 respondents in Kuala Lumpur and Selangor, 995 (84%) reported having unused medicines [2]. This might also lead to secondary issues; the improper disposal of medicine will lead to adverse effects on both the environment and human health. Without proper knowledge of the potential consequences, they often dispose of expired medications incorrectly, by either throwing them to the rubbish bin or flushing it in toilet bowl, which can have harmful effects on the environment [3].

Current applications focus primarily on medication reminders so that patient will adhere to their prescriptions. On the other hand, government initiatives provide general information on safe disposal methods and locations, but these are not combined with digital tools that remind users about their medications. In addition, these apps often neglect the importance of privacy of medication records, which will leave sensitive medical information vulnerable to unauthorized access. Disclosure of such sensitive information could compromise the privacy of individuals and potentially lead to misuse of data or other malicious purposes.

MediCare+ is a mobile application proposed as a solution for both medication adherence and disposal of unused or expired medicines. It will remind patients to take their medications as prescribed and provide

information on medicine disposal. The application will include a disposal map to help users find the nearest disposal sites, such as clinics, hospitals and pharmacies. Moreover, MediCare+ will also implement multifactor authentication and database encryption via 256-bit Advanced Encryption Standard (AES) to ensure privacy of the user data.

With the development of MediCare+, it will support environmental protection by providing guidance on how to properly dispose medicine. It also aligns with government policy and initiatives which aimed to reduce pharmaceutical waste and pollution. In addition, by offering a medication reminder, it encourages better adherence of medication. This will lead to reduce of the number of unused and expired medicine. Last but not the least, MediCare+ will be the first mobile application to combine medication reminders with disposal guidance and AES.

2. Related Work

This section discusses literature review on topics related to pharmaceutical pollution, medicine disposal practice, and large language models. On top of that, comparisons of existing systems have also been made in this section.

2.1 Pharmaceutical Pollution

Pharmaceuticals drugs can be defined as nonbiological chemicals that used for diagnosis, treatment, alteration or prevention of diseases, health condition, as well as for modification of health conditions or the structural and functional state of the human body [4]. Teasdale further narrowed and elaborates pharmaceuticals in his work by defining Active Pharmaceuticals Ingredients (API), which are the core chemicals that serve as an active ingredient in medicinal products [5]. However, despite its high application, pharmaceuticals have generally been overlooked as a source of pollution.

Due to wide usage of pharmaceuticals in our daily lives, pharmaceuticals pollution is deteriorating. Research has conducted for the presence of selected nine pharmaceuticals, which is amoxicillin, caffeine, chloramphenicol, ciprofloxacin, dexamethasone, diclofenac, nitrofurazone, sulfamethoxazole, and triclosan in drinking water from Putrajaya, Malaysia [6]. In their research a total of 80 drinking water samples were collected from kitchen tap in residential area in Putrajaya. The results showed that all the investigated pharmaceuticals are found in drinking water samples, with caffeine has the highest concentration among those pharmaceuticals [6].

Most of the pharmaceutical molecules are designed to be water-soluble so that the human body can absorb those molecules. However, the release of pharmaceutical molecules that enter water bodies will generate adverse effects on aquatic organisms and human beings. Several studies shows that it will contribute to the emergence of antibiotic-resistant bacteria and increase the risk of drug misuse or accidental poisoning [7] [8]. Specific examples include synthetic oestrogen, which can feminize fathead minnows, causing the lake ecosystem to collapse [9] and antidepressants which have been found to alter and affect the behaviour of shrimp [10]. On the other hand, human exposure to these contaminants may lead to adverse health effects, such as respiratory disorders, cancers and reproductive issues [11].

2.2 Large language Model Based Agents

Large language models are a type of artificial intelligence (AI) which are a subset of deep learning. In today's context, it typically refers to Transformer language model that can be pre-trained on massive text data and fine-tune for specific purposes [12].

When these powerful models are used as the core reasoning engine for an autonomous system, they form what is known as an LLM agent. The term LLM agent refers to a specific subtype of the broader AI agent category. While a general AI agent is any system that perceives its environment and acts autonomously, an LLM agent distinctly utilizes a large language model as its core engine for reasoning, planning, and interaction. This provides advanced capabilities in natural language understanding and flexible problem-solving. Consequently, all LLM agents are AI agents, but they represent a specialized and powerful evolution within that category [13]

Before the existence of Transformer models, Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) were the standards for sequence-to-sequence modelling [13]. However, both RNN and CNN have faced challenges such as longer training time difficulties with capturing long-range dependencies. A much simpler approach has proposed, which just using attention mechanisms for training language model, and it becomes the foundation for the emergence of large language model [14].

The improvement of the architecture has made the training of language models more efficient. Jason Wei has defined emergent abilities as "an ability is emergent if it is not present in smaller models but is present in larger models", which is one of the most important aspects that sets apart large language model from previous language models [15]. In his work, he further elaborates that increasing the scale of language model, whether the model parameters or training computes, will generally improve the performance of the language model [15].

2.3 Study of Existing Applications

In this section, a short explanation and description of existing applications and websites in the market will be provided. There are a total of three similar applications and websites that are chosen for the comparison with proposed application, which are Medisafe [16], MyMediSAFE [17] and Medication Reminder [18]. Medisafe and Medication Reminder are mobile applications, while MyMediSAFE is a website. Unlike MyMediSAFE, all the mobile applications register, and sign-in functionality is supported and hence the users can create a personalized individual account. On top of that, Medisafe, Medication Reminder and MediCare+ offer options to add, edit and delete medicine reminders. However, only MediCare+ integrates a unique feature to notify users about expired medications, promoting proper disposal practices.

Out of all the applications, only MediCare+ has an AI chatbot that instantly answers questions from users regarding medications and their disposal. It also features a disposal map with locations and tracking, which allow users to locate the nearest disposal sites and obtain details disposal sites, such as address, email, telephone number. While MyMediSAFE offers disposal guidance, it is presented as static information. The AI-powered chatbot in MediCare+ represents a significant advancement by providing a dynamic, interactive, and personalized user experience, allowing users to ask context-specific questions about disposal and medication. On the other hand, all the applications, except MyMediSAFE, offer cloud data storage to ensure a seamless experience across devices. However, only MediCare+ and Medisafe included Advanced Encryption Standard (AES) to protect local data.

Medisafe features extra medical management features, including health measuring recording, appointment tracking, and storing the doctors' contact's details. While MediCare+ does not include these features, its focus on medication reminders, proper disposal guidance, and AI-driven interactions sets it apart as a comprehensive solution for medication and disposal management.

Table 1 *Comparison table of existing applications*

Feature	Medisafe [16]	MyMediSAFE [17]	Medication Reminder [18]	MediCare+
Platform	Mobile Application	Website	Mobile Application	Mobile Application
Register and Sign-in	Yes	No	Yes	Yes
Add medicine reminder	Yes	No	Yes	Yes
Delete medicine reminder	Yes	No	Yes	Yes
Edit Medicine reminder	Yes	No	Yes	Yes
AI chat bot with function calling	No	No	No	Yes
Disposal Map	No	Yes	No	Yes
Location Tracker	No	No	No	Yes
Disposal Guidance	No	Yes	No	Yes
Store reminders in cloud	Yes	No	Yes	Yes
Encrypts local data	Yes (AES)	No	No	Yes (AES)
Expired medicine reminder	No	No	No	Yes
Record health measurement	Yes	No	Yes	No
Appointment tracker	Yes	No	No	No
Record doctor's contact	Yes	No	No	No

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 528-547

3. Methodology

A project methodology is a structured approach to implementing Software Development Lifecycle (SDLC) which can be viewed as a series of tasks [19]. This section explains the use of prototype model as the project management methodology in MediCare+. and the activities that had been carried out in each phase. The prototyping model focuses on building a working prototype early in the development phase to gather user feedback, refine requirements, and make improvements. It involves performing analysis, design and implementation simultaneously to rapidly develop a simplified version of the proposed system [19].

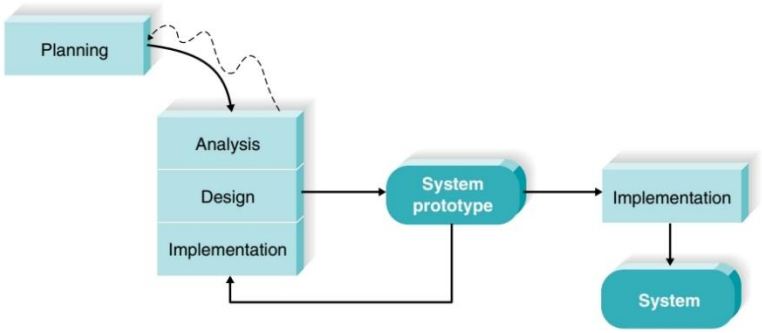


Fig. 1 *Prototyping Model [19]*

Fig. 1 shows the phases of prototyping, which includes planning, analysis, design, implementation and finalization phase. Each phase has its own assignment and output that need to produce during the entire project development, as shown in Table 2

Table 2 *System Development Workflow*

Phase	Task	Output
Planning	• Identifying dual problems of medication non-adherence and improper disposal • Determining the project scope and defining initial objectives.	• A project proposal outlining the concept for MediCare+ • A detailed project schedule (Gantt chart).
Analysis	• Gather user requirement • Determine system requirement • Identify hardware and software requirement	UML diagram which consists of: • Use case diagram • Sequence diagram • Activity diagram • Class diagram
Design	• Designing a responsive user interface • Structuring the NoSQL database schema for Firebase Firestore	• The complete Firestore database collection definitions • The final UI design for key application screens. •
Implementation	• Develop application • Test the application based on predefined test plan • Gather user feedback	• A functional, testable version of the MediCare+ application encompassing all modules. • A complete set of passed test results
System Finalization	• Develop finalized application	• Finalized version of MediCare+

4. Analysis and Design

This section outlines all the functional and non-functional requirements of the proposed application. It includes UML diagrams such as use case diagrams, activity diagrams, and class diagrams, along with database collections and use case specifications. Additionally, the section presents the interface design of the proposed application.

4.1 System Requirements

Requirement is a statement on the activities conducted by system or characteristics it need to possess [19]. After collecting and gathering data from planning phase, the system requirements need to be identified and determined. Table 3 shows the functional requirements of the system, whereas Table 4 shows the non-functionals.

Table 3 *Functional Requirement*

ID	Module	Requirement Description
F-01	Login	User can log in with a valid email and password.
F-02	Login	Application provides feedback for invalid or empty inputs.
F-03	Registration	A new user can register for an account.
F-04	Registration	Application validates input, such as checking for duplicate emails.
F-05	Medicine Reminder	User can add, edit, and delete medication reminders.
F-06	Medicine Reminder	Application sends notifications for medication doses and expiry dates.
F-07	Disposal Map	Display an interactive map with searchable disposal sites.
F-08	Disposal Map	Show detailed site information and provide actions (navigate, call, email).
F-09	Disposal Map	Use device location (with permission) to find nearby sites.
F-10	AI Chatbot	Act as a healthcare assistant for medication, health, and app-related questions.
F-11	AI Chatbot	Provide instructions on proper medicine disposal.
F-12	AI Chatbot	Use web search to provide cited, reliable information.
F-13	AI Chatbot	Find and display nearby disposal sites as interactive cards.

Table 4 *Non-functional Requirement*

ID	Category	Requirement Description
NF-01	Operational	The application must run on an Android device.
NF-02	Operational	The application requires an internet connection for cloud-based features.
NF-03	Performance	User interactions should complete within 5 seconds.
NF-04	Security	Access is restricted to authenticated users.
NF-05	Security	A strong password policy is enforced.
NF-06	Security	Sensitive user data is encrypted at rest in the cloud.

4.2 System Design

Fig. 2 shows the use case diagram of the proposed application. User will be the primary actor and hence will be at the left part of the use case diagram, whereas external system such as Firebase and Google Cloud will act as secondary actors and be placed at right. There are six main use case in the application which includes login, register, medication reminder, disposal guidance, disposal map and lastly AI chatbot.

Appendix A shows the sequence diagram of the proposed application, where it emphasizes the key interaction and its flow among the objects or entities. It consists of login sequence to check the username and password using Firebase Authentication and redirecting successful users to the home page. The registration sequence outlines the process of checking email uniqueness, verifying accounts, and notifying the user accordingly. The medication reminder sequence demonstrates how a user can add or modify a reminder or even delete a reminder, while inform a user regarding expiring medicine or upcoming reminders. Disposal map sequence will locate user's location and provide nearby disposal site to the user. Lastly, the AI chatbot sequence handles user queries about medications and disposal, retrieving grounded responses from Firebase and providing appropriate feedback.

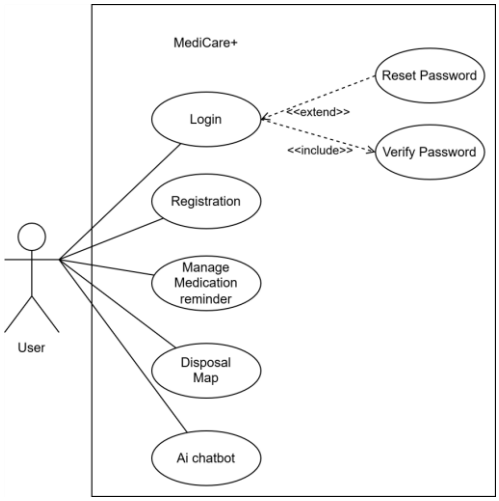


Fig. 2 *Use Case Diagram*

The application operation begins when the user opens the application. The user should complete registration before they can log in with email and password. Once validated, they are redirected to the homepage, else they are prompted to re-enter their credentials. On the homepage, users can access features such as medication disposal guidance, a disposal map showing the nearest disposal sites with location details and contact information, and adding, editing or deleting medication. Moreover, the AI chatbot can be used by the users to inquire about medications or disposal methods. Users can then either use or exit the application after navigating through these features. Fig. 3 shows the class diagram and Fig. 4 shows the activity diagram.

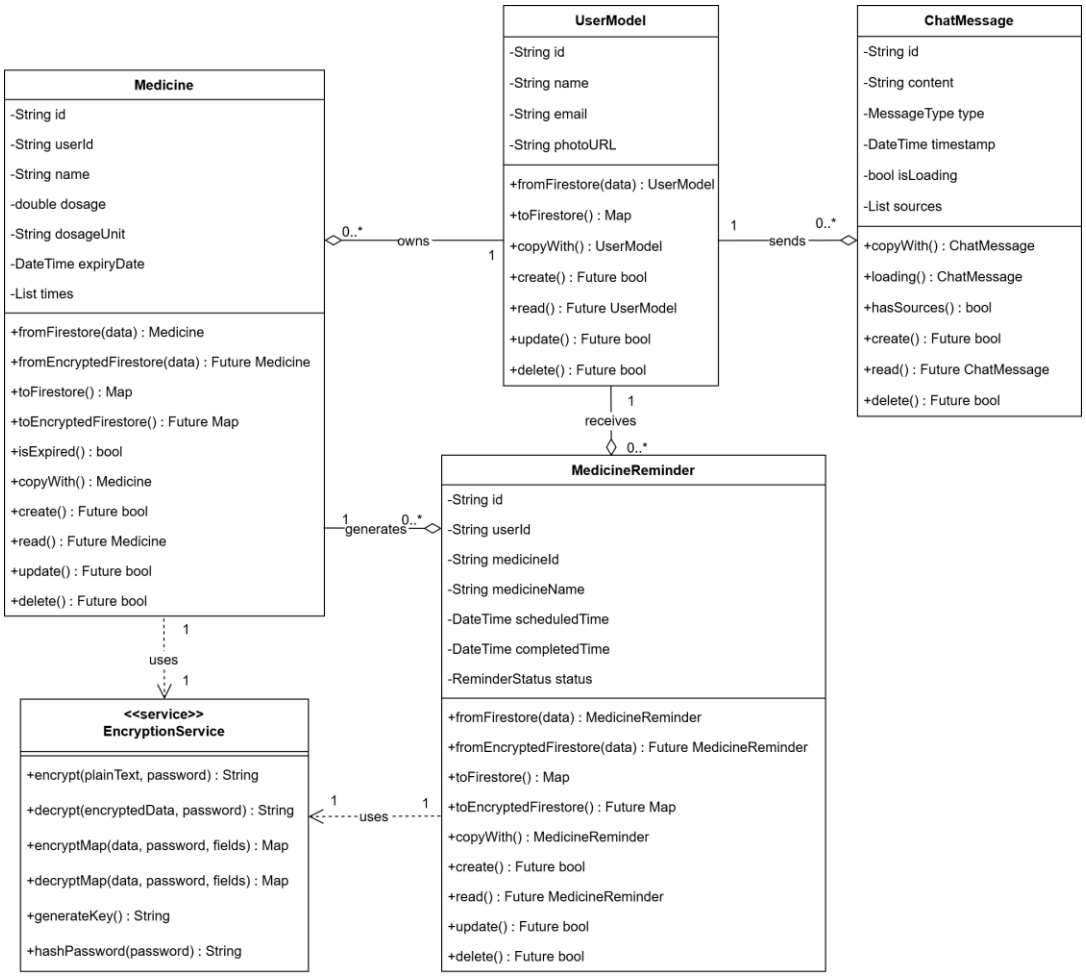


Fig. 3 *Class Diagram*

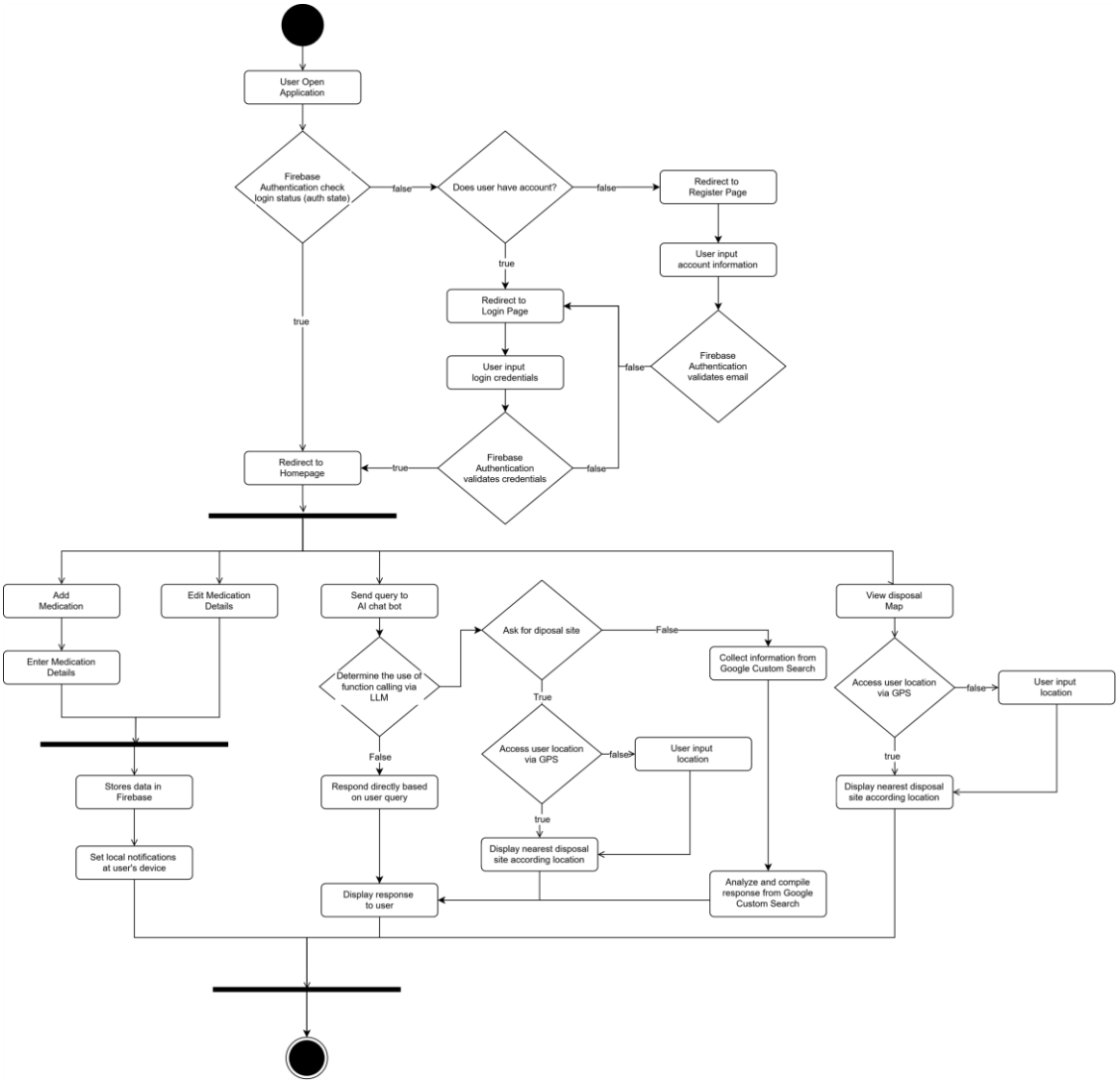


Fig. 4 *Activity Diagram*

Database design involves activities aimed at creating a structured database framework to efficiently store and manage end-user data [20]. In this application, the data at firebase will be stored in a series of collections, and hence there will be no schema [21]. Table 5 shows the collections created in Firebase.

Table 5 *Collections*

Collection	Field	Data Type
users	Id (obtain from Firebase Authentication)	String
	email	String
	name	String
medicines	Id	String
	userId	String
	name	String
	dosage	double
	dosageUnit	String
	repeat	int
	repeatUnit	String
	Times	List<TimeOfDay>

Table 5 *(cont.)*

Collection	Field	Data Type
medicines	expiryDate	Datetime
	createdAt	DateTime
medicineReminders	id	String
	userId	String
	medicineId	String
	medicineName	String
	dosageInfo	String
	scheduledTime	DateTime
	completedTime	DateTime
	status	enum
	createdAt	DateTime

5. Implementation and Testing

This section discusses the comprehensive implementation of MediCare+, encompassing all major modules including authentication, medicine management, AI chatbot, disposal mapping, and security enhancements. MediCare+ is structured according to Google's recommended clean architecture with several layers featuring distinct layers: a UI layer comprising widgets (View) and ViewModels, and a data layer (Model) that manages repositories and services [22]. Apart from that, the application follows Material 3 guidelines to ensures the consistency in colours and designs.

5.1 Login and Registration Module Implementation

The core of this module is the AuthService, which orchestrates the communication with Firebase. The registration process involves creating a new user account using their email and password as shown in Fig. 5(a). Upon successful creation in Firebase Authentication, a corresponding user profile document is created in the Firestore database to store application-specific information like their name.

The login process validates the user's credentials against Firebase as in Fig. 5(b). A critical step upon successful login is to securely cache the user's password in the device's secure storage for the session. This password is not stored permanently but is used to derive the key for encrypting and decrypting the user's local health data, ensuring that sensitive information remains protected on the device.

```dart
Future<User?> register({
  required String name,
  required String email,
  required String password,
}) async {
  try {
    UserCredential userCredential = await _auth
        .createUserWithEmailAndPassword(email: email, password: password);

    final user = userCredential.user;
    if (user != null) {
      final userModel = UserModel(
        id: user.uid,
        name: name,
        email: email,
        photoURL: 'assets/images/logo.png', // Default photo
      );
      await _userService.createUserData(userModel);

      // Securely cache the password to derive the encryption key for the new session
      await _cryptoManager.forceStorePassword(password);
    }
    return user;
  } catch (e) {
    return null;
  }
}
```

(a)

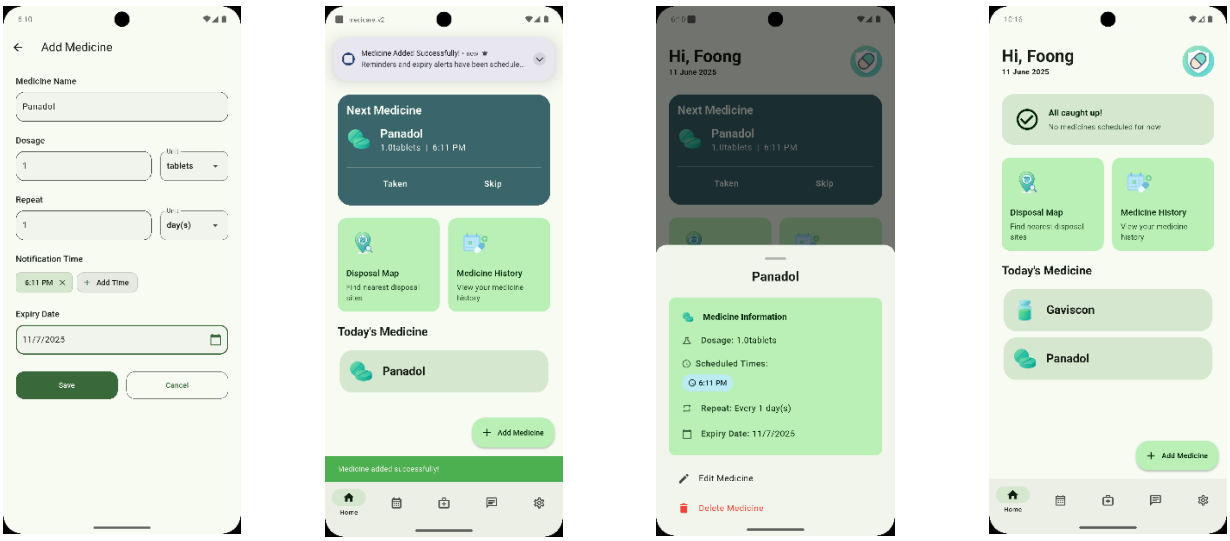
```dart
/// Logs in a user and caches the password for the encryption service.
Future<User?> login({required String email, required String password}) async {
  try {
    UserCredential userCredential = await _auth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );

    // On successful login, cache the password to unlock the user's encrypted data.
    if (userCredential.user != null) {
      await _cryptoManager.forceStorePassword(password);
    }
    return userCredential.user;
  } catch (e) {
    return null;
  }
}
```

(b)

Fig. 5 *Enhance password check (a) Register module(b) Login module*

5.2 Medicine Reminder Module Implementation

The medicine reminder modules provide comprehensive Create, Read, Update, Delete (CRUD) operations as display in Fig. 6. User can create new medicine entry via the Floating Action Button (FAB) located on the Home, Calendar and Medicine List page. Upon selecting a newly created medicine tile, a bottom sheet is displayed, presenting the details of that specific medication. From this interface, the user can then choose to either edit or delete the entry. Furthermore, the home page features a 'Next Medicine' card that displays complete information for the upcoming dose, including its name, dosage, and scheduled time. This interactive card allows the user to mark the medicine as either 'taken' or 'missed'.



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

Fig. 6 *(a) Add medicine page (b) Home page after adding medicine (c) Medicine detail page (d) Home page with cleared medicine*

Fig. 7 shows the notifications, it intelligently calculates and schedules two distinct types of local notifications, which are recurring dose reminders that fire precisely at user-defined times, and a separate series of expiry warnings. A key feature is the multi-stage expiry warning system, which automatically alerts the user 30, 7, 3, and 1 day(s) before a medicine expires, with a final notification on the expiry date itself.

```
Future<void> scheduleMedicineNotifications(Medicine medicine) async {

  await cancelMedicineNotifications(medicine.id);

  for (int timeIndex = 0; timeIndex < medicine.times.length; timeIndex++) {
    final time = medicine.times[timeIndex];

    final notificationDates = _calculateNotificationDates(
      medicine,
      time,
      startDate: DateTime.now(),
    );

    for (int dateIndex = 0; dateIndex < notificationDates.length; dateIndex++) {
      final notificationDate = notificationDates[dateIndex];
      // ... scheduling logic using notificationId, payload, etc. ...
    }
  }
}
```

```
Future<void> scheduleExpiryNotifications(Medicine medicine) async {
  final now = DateTime.now();
  final expiryDate = medicine.expiryDate;

  // Do not schedule if the medicine has already expired
  if (expiryDate.isBefore(now)) return;

  final List<DateTime> expiryNotificationDates = [];
  final warningDays = [30, 7, 3, 1];

  for (final days in warningDays) {
    final warningDate = expiryDate.subtract(Duration(days: days));
    if (warningDate.isAfter(now)) {
      expiryNotificationDates.add(warningDate);
    }
  }
  expiryNotificationDates.add(expiryDate);

  for (int index = 0; index < expiryNotificationDates.length; index++) {
    // ... logic to call _scheduleExpiryNotification with a different channel/style ...
  }
}
```

(a) **(b)**

Fig. 7 *Code implementation of(a) Schedule medicine notification (b) Schedule expiry notification*

5.3 AI Chatbot Module Implementation

The MediChat feature is powered by an integration with Google's Gemini AI, implemented via the Firebase AI Logic client SDKs. Its principal innovation is a dynamic, AI-driven search system that leverages Gemini's function calling capabilities. This elevates MediChat beyond a conventional chatbot, allowing it to operate as an autonomous agent that can execute complex tasks, rather than just providing text-based answers. When a search is deemed necessary, the AI intelligently formulates focused, medical-specific search terms to retrieve the most relevant data, which is then synthesized into a response with source citations for reliability.

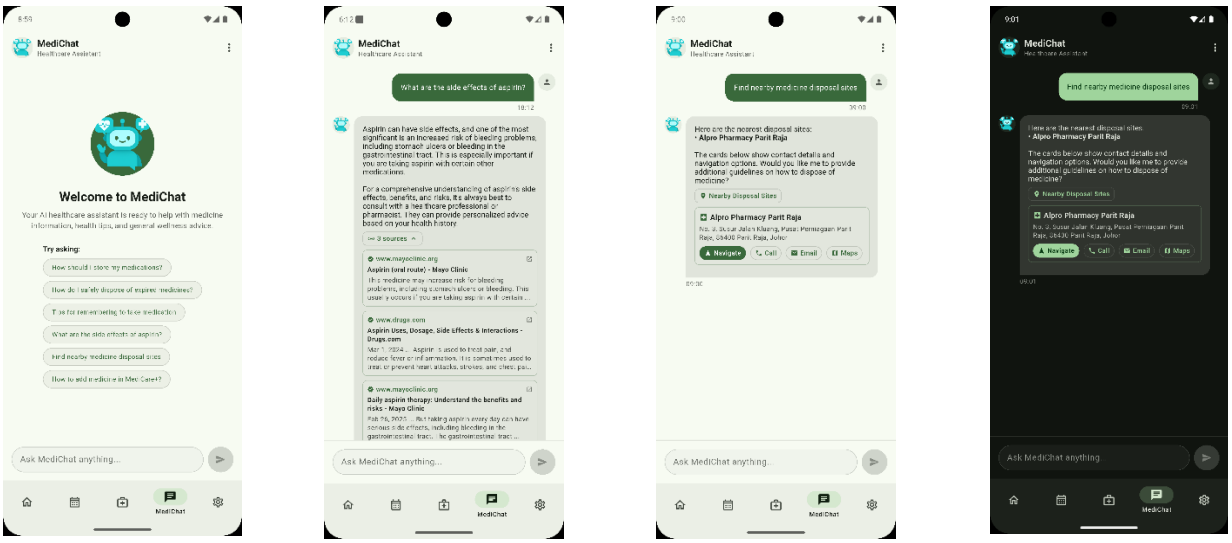
A key demonstration of this agency is its ability to find nearby medicine disposal sites (Fig. 8). When prompted, the AI can now execute a function that accesses the device's location services, determines the user's current position, and locates disposal sites within a customizable radius (defaulting to 5km). The findings are then presented as interactive site cards, from which the user can directly initiate navigation, a phone call, or an email, effectively bridging the gap between digital information and real-world action as in Fig. 9.

```dart
class ChatService {
  late final GenerativeModel _model;
  final SearchService _searchService = SearchService();

  void _initializeModel() {
    _model = FirebaseAI.googleAI().generativeModel(
      model: 'gemini-2.5-flash-preview-05-20',
      tools: [
        Tool.functionDeclarations([_createSearchFunctionDeclaration()]),
      ],
    );
  }

  FunctionDeclaration _createSearchFunctionDeclaration() {
    return FunctionDeclaration(
      'searchHealthInformation',
      'Search for reliable health and medical information from trusted sources',
      parameters: {
        'query': Schema.string(description: 'Specific search query'),
        'maxResults': Schema.integer(description: 'Maximum results'),
      },
    );
  }
}
```

```dart
FunctionDeclaration _createDisposalFunctionDeclaration() {
  return FunctionDeclaration(
    'findMedicineDisposalSites',
    'Find nearby medicine disposal sites with direct contact and navigation options when users ask about medication disposal.',
    parameters: {
      'userLocation': Schema.object(
        description:
            'User\'s location coordinates. If not provided, will attempt to get current location.',
        properties: {
          'latitude': Schema.number(
            description: 'Latitude coordinate of the user\'s location',
          ),
          'longitude': Schema.number(
            description: 'Longitude coordinate of the user\'s location',
          ),
        },
      ),
      'maxSites': Schema.integer(
        description:
            'Maximum number of disposal sites to return (default: 3, max: 5).',
      ),
      'radiusKm': Schema.number(
        description: 'Search radius in kilometers (default: 5.0, max: 50.0).',
      ),
    },
    optionalParameters: ['userLocation', 'maxSites', 'radiusKm'],
  );
}
```

(a) **(b)**

Fig. 8 *Code implementation of (a) AI initialization and Search function declaration (b) How AI handle function calls*

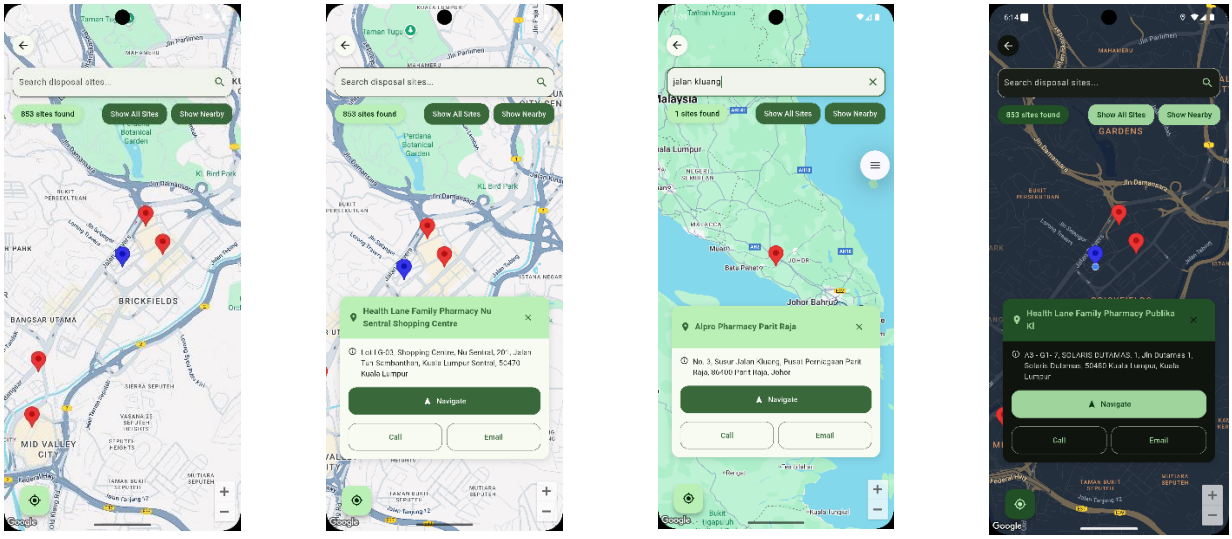


(a) **(b)** **(c)** **(d)**

Fig. 9 *(a) MediChat page (b) Search function called (c) Disposal function called (d) MediChat in Dark mode*

5.4 Disposal Map Implementation

MediCare+'s disposal map module in Fig. 10 provides users with an interactive map interface to locate authorized medicine disposal sites. Key functionalities include displaying all sites as custom markers, a text-based search, and a geospatial query to find nearby locations. This "nearby" feature works by obtaining the user's current location via the geolocator package and then using the Haversine formula to calculate the distance to each site. When a user taps a map marker, an info window appears, presenting site details and offering actionable buttons that leverage the url_launcher package to initiate a phone call, compose an email, or launch turn-by-turn navigation in an external map's application (Fig. 11).



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

Fig. 10 *(a) Disposal Map page (b) Info windows of disposal site (c) Text-based search (d) Disposal Map page in dark mode*

```dart
/// Get disposal sites within a certain radius (in kilometers) from a given location
Future<List<DisposalSite>> getDisposalSitesNearby({
  required double latitude,
  required double longitude,
  double radiusKm = 50.0,
}) async {
  final response = await getDisposalSites();
  return response.locations.where((site) {
    // Calculate distance using the Haversine formula
    final distance = _calculateDistance(
      latitude,
      longitude,
      site.latitude,
      site.longitude,
    );
    return distance <= radiusKm;
  }).toList();
}

/// Calculate distance between two coordinates using Haversine formula
double _calculateDistance(double lat1, double lon1, double lat2, double lon2) {
  const double earthRadius = 6371; // Earth's radius in kilometers
  final double dLat = _degreesToRadians(lat2 - lat1);
  final double dLon = _degreesToRadians(lon2 - lon1);
  final double a = (sin(dLat / 2) * sin(dLat / 2)) +
      cos(_degreesToRadians(lat1)) *
          cos(_degreesToRadians(lat2)) *
          (sin(dLon / 2) * sin(dLon / 2));
  final double c = 2 * atan2(sqrt(a), sqrt(1 - a));
  return earthRadius * c;
}
```

(a)

```dart
/// Get nearby disposal sites
Future<void> getNearbyDisposalSites({double radiusKm = 50.0}) async {
  // Ensure user's current location is available before proceeding
  if (_currentPosition == null) {
    _setErrorMessage('Current location not available');
    return;
  }

  _setLoading(true);
  _setErrorMessage(null);

  try {
    // Call the service to get sites within the specified radius
    final nearbyDisposalSites = await _disposalService.getDisposalSitesNearby(
      latitude: _currentPosition!.latitude,
      longitude: _currentPosition!.longitude,
      radiusKm: radiusKm,
    );
    _disposalSites = nearbyDisposalSites;
    _setMarkers(); // Update the markers on the map
  } catch (e) {
    _setErrorMessage('Failed to get nearby disposal sites: ${e.toString()}');
  } finally {
    _setLoading(false);
  }
}
```

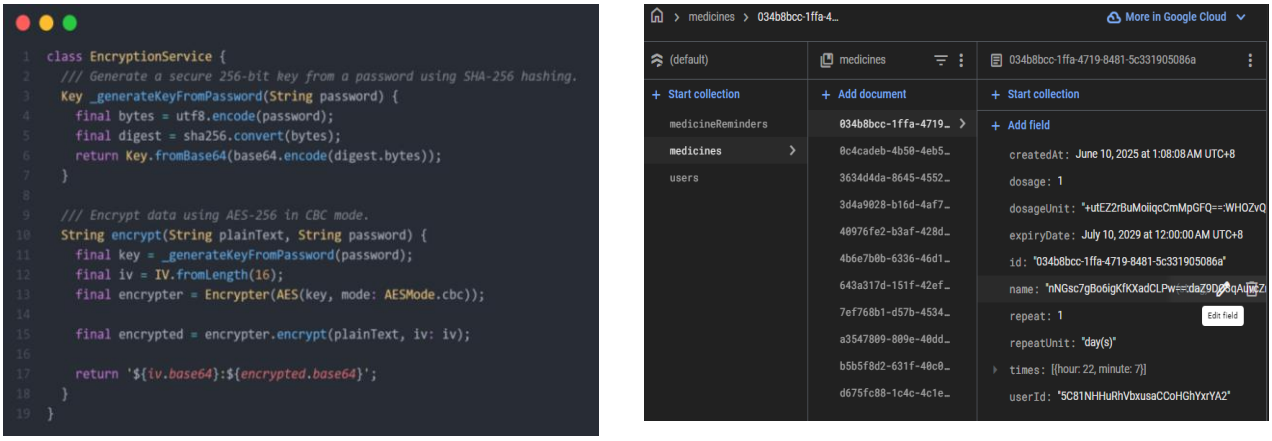
(b)

Fig. 11 *Code implementation of (a) Geospatial query with Haversine formula (b) Get nearby site*

5.5 Data Encryption Implementation

MediCare+ employs a data encryption to protect sensitive user data, such as medicine details, both on the device and in the cloud (at rest) as display in Fig. 12. The core of this system is a transient, session-based encryption key derived directly from the user's login password. Upon successful authentication, the password is used to generate a 256-bit AES key via a SHA-256 hash. This password is then temporarily cached for a 3-day period within the device's native secure enclaves (Keystore for Android) using flutter_secure_storage.

If the session expires or the user logs out, the cached password is wiped, and the encryption key will be inaccessible until the user re-authenticates. All cryptographic operations use the industry-standard AES-256 algorithm in CBC mode, with a unique, randomly generated Initialization Vector (IV) for each piece of encrypted data to ensure robust security. Before any sensitive data is sent to Firestore, a central CryptoManager selectively encrypts the necessary fields, ensuring that plain-text health information never leaves the user's device.



(a) **(b)**

Fig. 12 *Code implementation of (a) Encryption service (b) Data in Firestore*

5.6 Test Plan Result

Test plan can be defined as a series of test that will be conducted [18]. It consists of strategy or a set of specific test case to examines the system and defines an expected output. The developers can test the system by comparing the expected output and the actual results observed. Table 6 shows the result of the test plan for user using the proposed application.

Table 6 *Test Plan Result*

Module	Test Case	Requirement Tested	Expected Result	Test Result (Pass/Fail)
Login	Validate successful login with valid credentials	F-01, NF-04	User is successfully authenticated and redirected to the home page	PASS
	Attempt login with an invalid email format.	F-02	Display “Enter a valid email” error	PASS
	Attempt login with a correct email but an incorrect password.	F-02, NF-04	Display “Please check your credentials” error	PASS
	Attempt login with empty email and password fields.	F-02	Display “Email and Password required” error	PASS
Registration	Validate successful login with valid credentials	F-03	User account is created, and the user is redirected to the login page	PASS
	Attempt to register with an email that is already in use.	F-02	Display “Registration Failed” error	PASS
	Attempt to register with a password missing a special character.	NF-05	Display “Password must contain at least one special character” error	PASS
	Attempt to register with a password that is too short (<8 characters).	NF-05	Display “Password must be at least 8 character long” error	PASS
Medicine Reminder	Add a new medication reminder with all required details.	F-05	The reminder is saved, and the new medication appears on the home page and medicine list	PASS
	Edit the dosage and time for an existing medication reminder.	F-05	The details for the selected medication are successfully updated and reflected in the system	PASS

Table 6 *(cont.)*

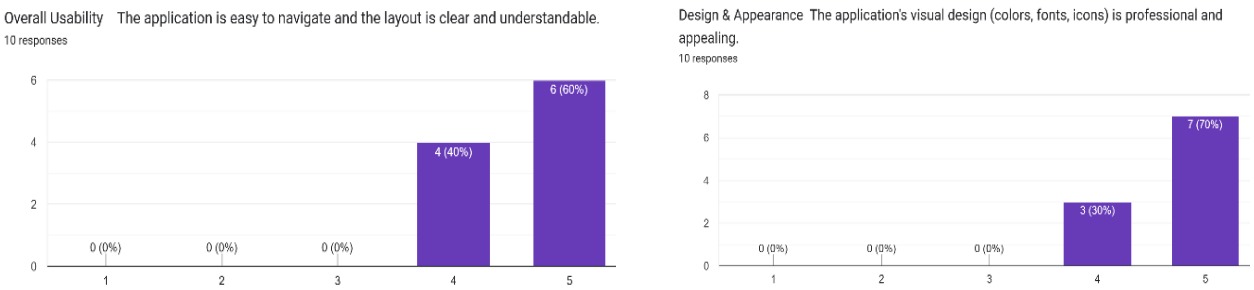
Module	Test Case	Requirement Tested	Expected Result	Test Result (Pass/Fail)
	Delete an existing medication reminder.	F-05	The selected medicine is permanently removed from the user's list and schedule.	PASS
	Validate that a dose reminder notification is triggered at the scheduled time.	F-06	A system notification appears at the correct time with accurate medication details.	PASS
	Validate that an expiry date warning is triggered for a medication nearing its expiry date.	F-06	A notification is received displaying "<Medicine> has expired" or "<Medicine> expired in <number> days"	PASS
Disposal Map	Open the Disposal Map and grant location permission.	F-0 7, F09	The map displays nearby disposal sites marked with red icons	PASS
	Open the Disposal Map and deny location permission.	F-09	The map loads, but a message is displayed indicating location permission is required to find nearby sites	PASS
	Use the search bar to find a specific disposal site by name.	F-07	The map updates to show markers matching the search query	PASS
	Click on a disposal site marker.	F-08	An info window appears showing the correct address, email, and phone number for the site	PASS
	Click the "Directions" icon in the info window.	F-08	The user is redirected to an external maps application for turn-by-turn navigation	PASS
AI Chatbot	Ask the chatbot, "What are the side effects of Paracetamol?"	F-10, F-12	The bot responds with accurate information and includes source citations from its web search	PASS
	Ask the chatbot, "How do I dispose of expired medicine?"	F-11	The bot provides the proper, step-by-step disposal methods based on its grounded facts	PASS
	Ask the chatbot, "Find nearby medicine disposal sites."	F-13	The bot responds with a list of nearby sites presented as interactive cards with navigation and contact options	PASS
Operational and Performance	Install and run the application on a target Android device.	NF-01	The application installs and launches successfully without crashing.	PASS
	Disable the internet connection and attempt to use a cloud-dependent feature (e.g., AI Chatbot).	NF-02	The application handles the lack of connection gracefully and displays a network error message.	PASS
	Measure the response time for loading the home screen and for AI chatbot queries under a stable network.	NF-03	Response times are consistently under the 5-second threshold.	PASS
Data Encryption and Persistence	Create a new medicine entry and inspect the corresponding record in the Firestore database.	NF-06	Sensitive fields (e.g., medicine name) in the database are stored as unreadable, encrypted strings	PASS
	Create a medicine, log out, and then log back in with the same account.	F-05, NF06	The previously added medicine data is present, decrypted, and correctly displayed in the app.	PASS

5.7 User Acceptance Test Result

User acceptance test is the test in which users test the system using real data [19]. It consists of strategy or a set of specific test case to examines the system and defines an expected output. The primary goal of this test is to validate the application's workflow and usability from the user's perspective. For this project, a group of target users was tasked with performing a series of predefined tasks, and their feedback was collected to determine if the application is intuitive, effective, and ready for deployment as in Fig. 13(a)(b)-. The test involved a diverse group of 10 participants. The respondents are evenly distributed by gender and consist of wide range of age group. The largest segment, at 40%, was from the 25-55 age bracket. This was followed by the 18-24 group (30%), the 55 and above group (20%), and the Below 18 group (10%).

The selection of a demographically diverse group of participants was a deliberate strategy to validate the usability and accessibility of MediCare+ across its entire potential user base. Medication management is a universal need that affects individuals of all ages, each with different technological fluencies and user experience expectations. For example, tech-native users (18-55) helped validate the core functionality and modern interface, while the 55+ group, a key demographic, provided essential feedback on accessibility, such as font legibility and navigational clarity. By ensuring positive feedback across these varied segments, we can be more confident that MediCare+ is not just functional, but truly effective and user-friendly for any individual who needs it.

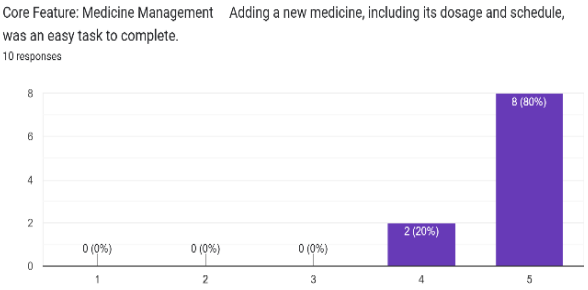
The overall user experience and design of the MediCare+ application were met with strong approval. All respondents found the application easy to use, with 100% of users rating the application's navigation and layout clarity positively. A significant 60% (6 respondents) gave the highest rating of 5, and 40% (4 respondents) gave a rating of 4, which indicates an intuitive and user-friendly interface. Furthermore, the visual design was highly praised, as 70% (7 respondents) rated the application's colours, fonts, and icons a 5, while the remaining 30% (3 respondents) rated it a 4. This combined feedback confirms that the application has both a professional, appealing visual presentation and a highly functional user interface.



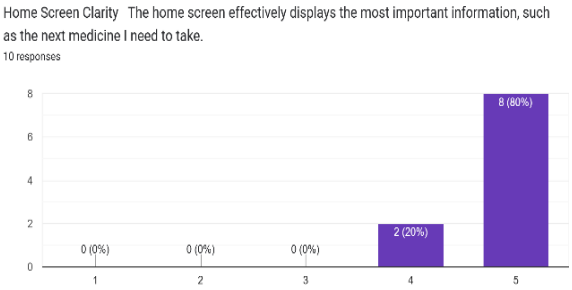
(a) **(b)**

Fig. 13 *Result of(a) Overall usability (b) Design & appearance*

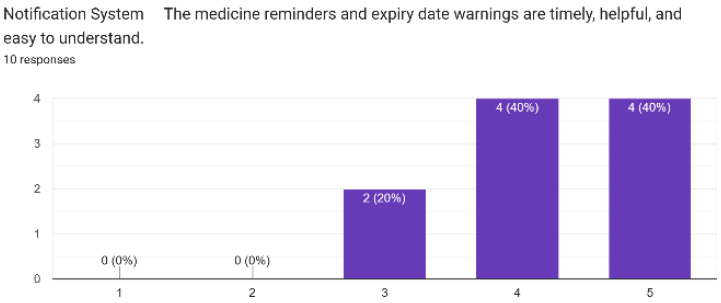
The core functionalities designed for daily medication management were tested and received excellent feedback from users (Fig. 14(a)-(c)). The task of adding a new medicine was perceived as very easy, with 80% (8 respondents) giving this feature a top rating of 5 and 20% (2 respondents) rating it a 4, demonstrating the efficiency of this core function. Similarly, the home screen was deemed effective at displaying essential information, as 80% (8 respondents) strongly agreed (rating of 5) that it clearly shows the next medicine to be taken. The notification system for reminders and expiry warnings was also positive, with 40% (4 respondents) rating it a 5 and another 40% rating it a 4. However, a small portion, 20% (2 respondents), gave a neutral rating of 3, suggesting that while the system is helpful, it may be an area for minor refinement to meet all users' preferences perfectly.



(a)



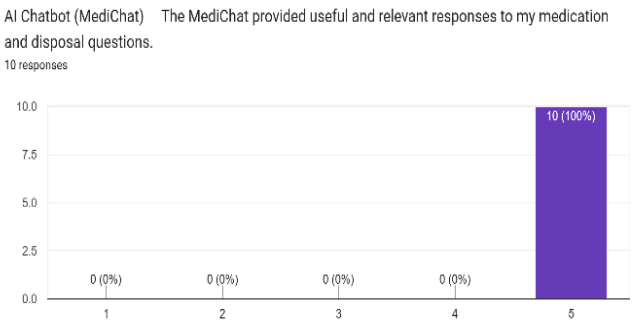
(b)



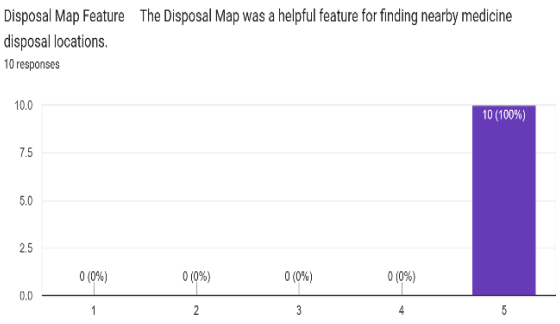
(c)

Fig. 14 *(a)Medicine management (b) Home screen clarity (c)Notification system*

The application's unique features, the AI Chatbot (MediChat) and the Disposal Map, were unanimously well-received and stood out as exceptionally successful components of the application. Fig. 15(a) show that the AI Chatbot was a clear success, with all 10 respondents (100%) giving it a perfect rating of 5, indicating that it consistently provided useful and relevant answers to medication and disposal questions. In the same vein, the Disposal Map feature in Fig. 15(b) received a perfect score from all 10 respondents (100%), who rated it a 5, confirming that it is a highly helpful and valued tool for finding nearby medicine disposal locations.



(a)



(b)

Fig. 15 *Result of(a) AI chatbot (b) Disposal Map*

The overall satisfaction level in Fig. 16 reflects the positive feedback on individual features. When asked if they would find the application useful for managing their medications, 100% of users responded positively. 60% (6 respondents) expressed strong agreement (rating of 5), and 40% (4 respondents) expressed agreement (rating of 4).

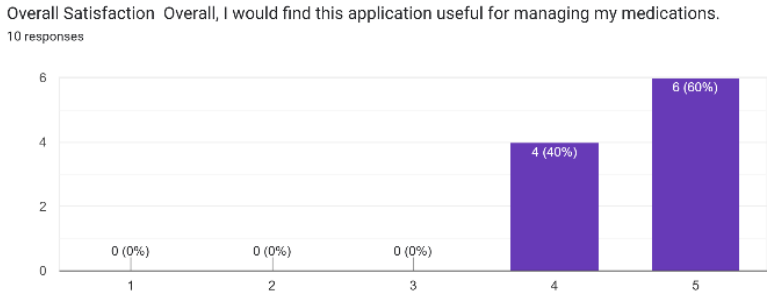


Fig. 16 *Result of overall satisfaction*

In addition to the quantitative ratings, respondents provided specific qualitative feedback that offers actionable suggestions for enhancement as in Fig. 17. For instance, the suggestion to "make notification more louder" provides valuable context for the quantitative data on the Notification System, where 20% of users gave a neutral score. This feedback indicates that the effectiveness of alerts could be improved by increasing the default volume of notification sounds, a critical factor for ensuring users do not miss important medication reminders. Another key piece of feedback was that "Maybe the word in medicine details can be larger." This suggestion points towards an important accessibility improvement. While the overall design was praised, this comment highlights a need to increase the font size within the "medicine details" section to enhance readability, particularly for older users or those with visual impairments, who are a key demographic for a medication management application.

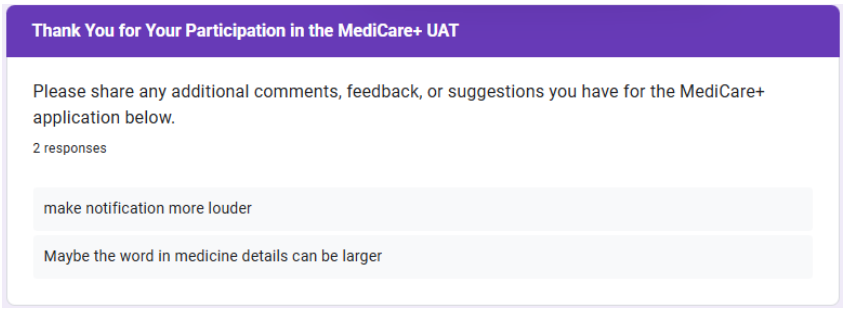


Fig. 17 *Feedback and Suggestions*

6. Conclusion

Medication non-adherence and the improper disposal of pharmaceuticals pose significant challenges to public health and the environment. To address these deficiencies, the MediCare+ application was developed as a cohesive platform that integrates medication reminders, secure data handling, and AI-driven disposal guidance.

The MediCare+ application offers several key advantages that distinguish it from existing solutions. It enhances security through a session-based AES-256 encryption system to protect sensitive data, a measure not universally adopted by current health apps. The application provides a comprehensive reminder system featuring multi-stage expiry warnings, directly addressing a gap in standard reminder tools. The core innovation is the MediChat AI agent, which utilizes Google's Gemini with Function Calling. This allows it to perform advanced tasks, such as grounding responses with live web searches or finding nearby disposal sites using GPS, positioning it as a sophisticated healthcare assistant rather than a conventional chatbot. These features, particularly the integration of an interactive disposal map with an AI agent, collectively address the fragmented nature of existing medication management tools.

However, some limitations in the current implementation are acknowledged. Firstly, the medicine disposal site data is loaded from a static local file, which cannot reflect real-time changes. Secondly, the AI's web search capability relies on a free-tier Google Custom Search API with a daily query limit. Lastly, while built with Flutter, development and testing were primarily concentrated on the Android platform, leaving the iOS version without full optimization.

To address these limitations, several enhancements are proposed for future iterations. First, the disposal map could be integrated with a live government or partner API to ensure location data is always current. Furthermore, the application's functionality could be expanded by enhancing the AI's agentic capabilities to include tasks such as scheduling pharmacy consultations for medication reviews. Finally, a dedicated testing and release cycle for iOS is recommended to ensure a fully optimized cross-platform experience, allowing for wider user feedback to further refine the application's features and usability.

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 interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *C. W. K. Foong, S. N. Ramli;* ***data collection:*** *C. W. K. Foong, S. N. Ramli;* ***analysis and interpretation of results*** *C. W. K. Foong, S. N. Ramli;* ***draft manuscript preparation:*** *C. W. K. Foong, S. N. Ramli. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Y. Peng *et al.*, "Effectiveness of Mobile Applications on Medication Adherence in Adults with Chronic Diseases: A Systematic Review and Meta-Analysis," *Journal of Managed Care & Specialty Pharmacy*, vol. 26, no. 4, pp. 550–561, Mar. 2020, doi: 10.18553/jmcp.2020.26.4.550.

[2] L. S. Wang, Z. Aziz, and Z. Chik, "Disposal practice and factors associated with unused medicines in Malaysia: a cross-sectional study," *BMC Public Health*, vol. 21, no. 1, Sep. 2021, doi: 10.1186/s12889-021-11676-x.

[3] Alpro Pharmacy, "Disposing of excess medications safely is essential for our health - Alpro Pharmacy," *Alpro Pharmacy*, Jan. 30, 2023. https://www.alpropharmacy.com/news_post/disposing-of-excess-medications-safely-is-essential-for-our-health/

[4] C. G. Daughton and T. A. Ternes, "Pharmaceuticals and personal care products in the environment: agents of subtle change?," *Environmental Health Perspectives*, vol. 107, p. 907, Dec. 1999, doi: 10.2307/3434573.

[5] A. Teasdale, D. Elder, and R. W. Nims, *ICH Quality Guidelines: An Implementation Guide*. John Wiley & Sons, 2017.

[6] S. M. Praveena, M. Z. M. Rashid, F. A. M. Nasir, W. S. Yee, and A. Z. Aris, "Occurrence and potential human health risk of pharmaceutical residues in drinking water from Putrajaya (Malaysia)," *Ecotoxicology and Environmental Safety*, vol. 180, pp. 549–556, May 2019, doi: 10.1016/j.ecoenv.2019.05.051.

[7] E. M. Wellington *et al.*, "The role of the natural environment in the emergence of antibiotic resistance in Gram-negative bacteria," *The Lancet Infectious Diseases*, vol. 13, no. 2, pp. 155–165, Jan. 2013, doi: 10.1016/s1473-3099(12)70317-1.

[8] S. D. Paul and S. Gandhi, "Fate of Improper Drug Disposal and its Impact on Health," *International Journal of Pharmaceutical Sciences Review and Research*, vol. 64, no. 1, pp. 188–193, Sep. 2020, doi: 10.47583/ijpsrr.2020.v64i01.034.

[9] K. A. Kidd *et al.*, "Collapse of a fish population after exposure to a synthetic estrogen," *Proceedings of the National Academy of Sciences*, vol. 104, no. 21, pp. 8897–8901, May 2007, doi: 10.1073/pnas.0609568104.

[10] Y. Guler and A. T. Ford, "Anti-depressants make amphipods see the light," *Aquatic Toxicology*, vol. 99, no. 3, pp. 397–404, Jun. 2010, doi: 10.1016/j.aquatox.2010.05.019.

[11] V. Chander *et al.*, "Pharmaceutical compounds in drinking water," *Journal of Xenobiotics*, vol. 6, no. 1, Jun. 2016, doi: 10.4081/xeno.2016.5774.

[12] W. X. Zhao *et al.*, "A survey of large language models," *arXiv.org*, Mar. 31, 2023. https://arxiv.org/abs/2303.18223

[13] Zhao, P., Jin, Z., & Cheng, N. (2023, September 23). *An in-depth survey of large language model-based artificial intelligence agents*. arXiv.org. https://arxiv.org/abs/2309.14365

[14] A. Vaswani *et al.*, "Attention is All you Need," *arXiv (Cornell University)*, vol. 30, pp. 5998–6008, Jun. 2017, [Online]. Available: https://arxiv.org/pdf/1706.03762v5

[15] J. Wei *et al.*, "Emergent abilities of large language models," *arXiv.org*, Jun. 15, 2022. https://arxiv.org/abs/2206.07682

[16] "Medisafe Pill & Med Reminder - Apps on Google Play." https://play.google.com/store/apps/details?id=com.medisafe.android.client&hl=en-US

[17] "MyMediSAFE: Ministry of Health, Malaysia | Medication Disposal." https://www.mymedisafe.org.my/
[18] "Medication Reminder - apps on Google Play." https://play.google.com/store/apps/details?id=com.makio.medica
[19] A. Dennis, B. H. Wixom, and R. M. Roth, *Systems analysis and design*, 8th ed. John Wiley & Sons, 2022.
[20] C. Coronel, S. Morris, and P. Rob, *Database Systems: design, implementation, and management*, 13th ed. Cengage Learning, 2019.
[21] Firebase, "Cloud Firestore Data Model," *Firebase*, Dec. 2024. https://firebase.google.com/docs/firestore/data-model (accessed Dec. 09, 2024).
[22] Flutter. (2024, September). *Flutter Architectural Overview*. Retrieved December 9, 2024, from https://docs.flutter.dev/resources/architectural-overview

Appendix A: Gantt Chart

Fig. A.1 *Gantt Chart*

Appendix B: Overall Sequence Diagram

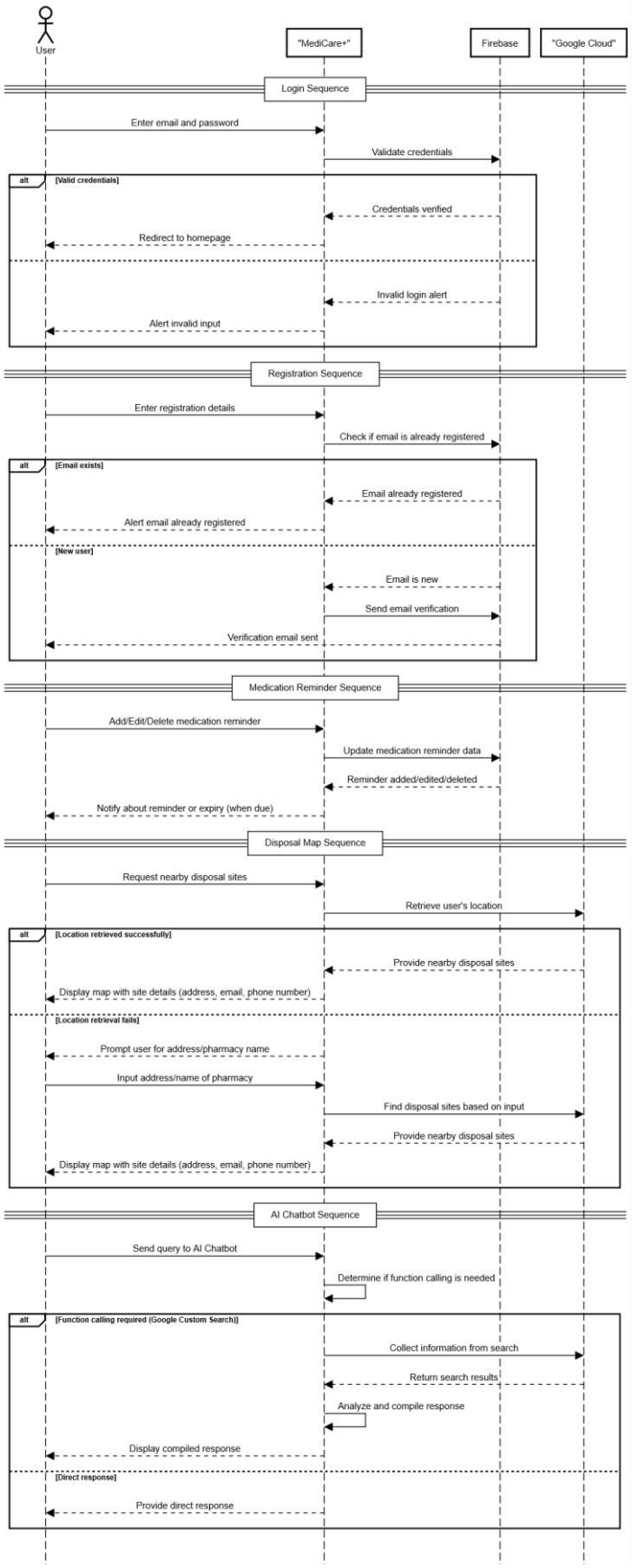


Fig. A.2 *Sequence Diagram*