aayu-mitra: Simplified Guide (v2)

Context-Aware LLM Assistant with Flutter App

This guide outlines the architecture for "aayu-mitra," a context-aware elderly care assistant. This version **focuses on the core product:** a Raspberry Pi assistant (Pipeline A) that communicates in real-time with a caregiver's Flutter app, using Firebase as the backend.
The advanced ML anomaly detection (Pipeline B) is considered "Future Work" for this phase.

1. Project Overview

The system has two main components:

1. **The Pi Hub (Python CLI App):**
    - A Python script running on the Raspberry Pi.
    - Listens for the user's voice (using Vosk).
    - Fetches the user's profile (medication, routines) from Firebase to build a "context."
    - Sends the user's query + context to the Gemini LLM for a smart, personalized response.
    - Logs the conversation to the messages collection.
    - Generates alerts in Firestore if the user asks for help (e.g., "I feel dizzy").
    - Listens for real-time messages from the caregiver (from the messages collection) and speaks them aloud.
2. **The Caregiver App (Flutter):**
    - The Flutter app shown in your screenshots.
    - Listens in real-time to the alerts collection to show notifications.
    - Listens to the messages collection to show the "Conversation History."
    - Allows the caregiver to WRITE data to Firebase (add medication, routines, health records, or send a chat message).

2. Simplified API Contract (Firebase Schema)

This is the data structure your Python and Flutter apps **MUST** follow.
Root Path: artifacts/{appId}/public/data/

Endpoint: /user_profile

- **Document ID:** {pi_id} (e.g., "pi-hub-001")
- **Data Model: UserProfile**

```
{
  "name": "string",
  "age": "number",
  "notes": "string",
  "sleep_schedule": {
    "planned_bedtime": "string (HH:MM)",
    "planned_wakeup": "string (HH:MM)"
```

```
  }
}
```

Endpoint: /medication

- **Document ID:** Auto-generated ({medication_id})
- **Data Model: Medication**

```
{
  "pi_id": "string", // Foreign key to user_profile
  "title": "string",
  "category": "string",
  "amount": "string",
  "time_slots": "array<string>",
  "repeat_days": "array<string>"
}
```

Endpoint: /routines

- **Document ID:** Auto-generated ({routine_id})
- **Data Model: Routine**

```
{
  "pi_id": "string", // Foreign key
  "title": "string",
  "time": "string (HH:MM)",
  "repeat_days": "array<string>",
  "notes": "string"
}
```

Endpoint: /reminders

- **Document ID:** Auto-generated ({reminder_id})
- **Data Model: Reminder**

```
{
  "pi_id": "string", // Foreign key
  "label": "string",
  "details": "string",
  "date": "string (YYYY-MM-DD)",
  "time": "string (HH:MM)",
  "priority": "string ('Low', 'Normal', 'High')",
  "completed": "boolean"
}
```

Endpoint: /health_records

- **Document ID:** Auto-generated ({record_id})
- **Data Model: HealthRecord**

```
{
  "pi_id": "string", // Foreign key
  "date": "timestamp",
  "weight_kg": "number",
  "blood_pressure": "string",
  "heart_rate_bpm": "number",
  "temperature_f": "number"
}
```

Endpoint: /alerts

- **Document ID:** Auto-generated ({alert_auto_id})
- **Data Model: Alert**

```
{
  "pi_id": "string", // Foreign key
  "level": "string ('high', 'info')",
  "source": "string ('Pipeline_A_LLM')",
  "message": "string",
  "timestamp": "timestamp",
  "acknowledged": "boolean"
}
```

Endpoint: /messages

- **Document ID:** Auto-generated ({message_auto_id})
- **Data Model: Message**

```
{
  "pi_id": "string", // Foreign key
  "sender": "string ('Mitara', 'Elderly', 'Caregiver')",
  "text": "string",
  "timestamp": "timestamp"
}
```

3. Code Examples: Reading & Writing Data

This is how your two apps will communicate using the API Contract.

Python CLI App (Raspberry Pi)

Role 4 & 5 (Hardware/ML, Pi Assistant)

```
# --- Setup (Run Once) ---
import firebase_admin
from firebase_admin import credentials, firestore
```

```
# Use your service account JSON file
cred = credentials.Certificate("path/to/your/serviceAccountKey.json")
firebase_admin.initialize_app(cred)
db = firestore.client()

# This is the root path from our contract
# NOTE: Omit {appId} if using the Admin SDK directly, or include your
specific ID.
# For simplicity, we'll assume the root is the collections.
PI_ID = "pi-hub-001" # This Pi's unique ID
APP_ID = "your-app-id" # Get this from your project setup
data_root = f"artifacts/{APP_ID}/public/data"


# --- WRITE: Send a new Alert (e.g., LLM detects distress) ---
def send_alert(message_text):
    alert_data = {
        "pi_id": PI_ID,
        "level": "high",
        "source": "Pipeline_A_LLM",
        "message": message_text,
        "timestamp": firestore.SERVER_TIMESTAMP,
        "acknowledged": False
    }
    db.collection(f"{data_root}/alerts").add(alert_data)
    print("Sent 'high' alert.")

# --- WRITE: Log a conversation turn ---
def log_message(sender_name, message_text):
    msg_data = {
        "pi_id": PI_ID,
        "sender": sender_name, # "Elderly" or "Mitara"
        "text": message_text,
        "timestamp": firestore.SERVER_TIMESTAMP
    }
    db.collection(f"{data_root}/messages").add(msg_data)

# --- READ: Get User Profile (to build LLM context) ---
def get_user_context():
    # Get user profile
    profile_ref =
db.collection(f"{data_root}/user_profile").document(PI_ID)
    profile = profile_ref.get().to_dict()

    # Get medication
    med_ref = db.collection(f"{data_root}/medication").where("pi_id",
"==", PI_ID)
```

```
    meds = [doc.to_dict() for doc in med_ref.stream()]

    context = f"User Profile: {profile}\nMedication: {meds}"
    return context

# --- LISTEN: Real-time listener for caregiver messages ---
def listen_for_messages():
    message_query = db.collection(f"{data_root}/messages") \
                      .where("pi_id", "==", PI_ID) \
                      .where("sender", "==", "Caregiver") \
                      .order_by("timestamp",
direction=firestore.Query.DESCENDING) \
                      .limit(1)

    def on_snapshot(doc_snapshot, changes, read_time):
        for change in changes:
            if change.type.name == 'ADDED':
                new_message = change.document.to_dict()["text"]
                print(f"New message from caregiver: {new_message}")
                # TODO: Call your eSpeak function to speak this text
                # os.system(f"espeak '{new_message}'")

    print("Listening for new messages from caregiver...")
    message_query.on_snapshot(on_snapshot)
```

Flutter App (Dart Code)

Role 2 (App Developer)

```
// --- Setup (Run Once in main.dart) ---
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_core/firebase_core.dart';

// Make sure to await Firebase.initializeApp() in your main()

// Get the Firestore instance
final FirebaseFirestore db = FirebaseFirestore.instance;
final String PI_ID = "pi-hub-001"; // The ID of the Pi this user is
monitoring
final String APP_ID = "your-app-id"; // Your app ID
final String dataRoot = "artifacts/$APP_ID/public/data";

// --- WRITE: Add a new Medication ---
Future<void> addMedication(String title, String category, List<String>
slots, List<String> days) async {
  final medicationData = {
    "pi_id": PI_ID,
```

```
    "title": title,
    "category": category,
    "amount": "2 pills", // Get from form
    "time_slots": slots,
    "repeat_days": days
  };

  await db.collection("$dataRoot/medication").add(medicationData);
  print("Medication added.");
}

// --- WRITE: Send a Message (from Conversation History screen) ---
Future<void> sendMessage(String messageText) async {
  final messageData = {
    "pi_id": PI_ID,
    "sender": "Caregiver",
    "text": messageText,
    "timestamp": FieldValue.serverTimestamp()
  };

  await db.collection("$dataRoot/messages").add(messageData);
  print("Message sent.");
}

// --- READ: Get Health Records (for "Track Health" screen) ---
Future<List<QueryDocumentSnapshot>> getHealthRecords() async {
  QuerySnapshot snapshot = await
db.collection("$dataRoot/health_records")
      .where("pi_id", "==", PI_ID)
      .orderBy("date", descending: true)
      .get();

  return snapshot.docs;
}

// --- LISTEN: Real-time listener for Alerts (for "Notifications"
screen) ---
// Use a StreamBuilder widget in your Flutter UI

Stream<QuerySnapshot> getAlertsStream() {
  // This Stream will update your UI automatically
  return db.collection("$dataRoot/alerts")
      .where("pi_id", "==", PI_ID)
      .orderBy("timestamp", descending: true)
      .snapshots();
}

/*
```

Example StreamBuilder in your widget:

```
StreamBuilder<QuerySnapshot>(
  stream: getAlertsStream(),
  builder: (BuildContext context, AsyncSnapshot<QuerySnapshot>
snapshot) {
    if (snapshot.hasError) {
      return Text('Something went wrong');
    }
    if (snapshot.connectionState == ConnectionState.waiting) {
      return CircularProgressIndicator();
    }

    // You have the new data! Build your list
    return ListView(
      children: snapshot.data!.docs.map((DocumentSnapshot document) {
        Map<String, dynamic> data = document.data()! as Map<String,
dynamic>;
        return ListTile(
          title: Text(data['message']),
          subtitle: Text(data['level']),
        );
      }).toList(),
    );
  },
)
*/
```