IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

Help Link: Design and Implementation of an AI-Powered Mobile Emergency Assistance Application Using Flutter and TensorFlow Lite

Shivam[1], Satyam Singh[2], Mrs. Vandana Sharma[3]
B.Tech CSE (AIML), 8th Semester[1,2]
Supervisor, Department of Computer Science and Engineering[3]
Sunder Deep Engineering College, Ghaziabad, U.P., India

Abstract: *Delayed access to emergency information during a medical crisis remains a leading cause of preventable mortality, particularly in regions with limited emergency infrastructure. Help Link is a cross-platform mobile application built in Flutter (Dart) that consolidates four critical emergency capabilities into a single, offline-capable system: (i) one-tap direct-dialling of fifteen pre-loaded national emergency services; (ii) on-device TensorFlow Lite injury classification with a configurable 80% confidence gate; (iii) GPS-driven discovery of nearby healthcare facilities with Haversine-formula distance computation and per-facility walking and driving time estimates; and (iv) step-by-step, numbered first-aid instructions optionally narrated via text-to-speech. The application follows a strict Model-View-ViewModel (MVVM) architectural pattern with constructor-injected services, enabling full unit testing without hardware dependencies. A custom three-breakpoint responsive layout engine adapts the interface across compact (< 600 dp), medium (600–1023 dp), and expanded (≥ 1024 dp) screen sizes. The healthcare discovery screen renders a rich facility card for each result, displaying facility type (Hospital / Urgent Care / Clinic), colour-coded type badge, address, capabilities chips (e.g., Emergency, 24hr, ICU, Surgery), distance in kilometres, estimated walking time at 5 km/h, estimated driving time at 30 km/h, and direct-call and turn-by-turn navigation actions. Because the TFLite model is bundled as a compiled application asset and all contact and first-aid data reside in-memory at runtime, the four core features function entirely without internet connectivity. This paper details the system architecture, MVVM design, ML inference pipeline, healthcare proximity engine, responsive layout system, and the engineering decisions underlying each module.*

Keywords: Flutter, TensorFlow Lite, MVVM, Emergency Response, On-Device ML Inference, First Aid, GPS, Haversine Distance, Text-to-Speech, Offline Mobile Application, Healthcare Facility Discovery.

I. INTRODUCTION

Emergency situations demand immediate, accurate, and cognitively accessible information at a moment when fear and panic impair rational decision-making. Medical research has long identified the 'golden hour' — the interval immediately following injury during which prompt intervention most dramatically improves survival outcomes and reduces long-term morbidity [1]. In India, this window is frequently missed: rural and semi-urban areas account for a disproportionate share of road-accident fatalities [2], and a significant fraction of these deaths are attributable not to the severity of injury alone, but to delayed or incorrect first-response. Internet connectivity — the assumption underlying most modern emergency applications — cannot be relied upon in precisely the environments where emergency assistance is most urgently needed.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

Help Link was developed to address this fragmentation directly. Rather than requiring a user under stress to recall an emergency number, open a browser, search for first-aid guidance, and separately navigate to a hospital, Help Link consolidates all four tasks into a single Flutter application with a home screen that presents exactly two cards: Injury Identification and Emergency Contacts. Every subsequent user action — selecting a contact to dial, photographing an injury, reading or listening to first-aid steps, or locating a nearby hospital — is reachable within two taps from the home screen.

The system is architected around three principles. First, offline-first: the TFLite model, all fifteen emergency contacts, and all six first-aid instruction sets are available with no network request. Second, one-tap critical paths: the flutter_phone_direct_caller plugin initiates a native call without the intermediary system dialler, and the 'Nearest Healthcare' button becomes visible only after a confident injury classification has been returned, guiding the user's next action. Third, progressive disclosure: the healthcare map screen reveals a scrollable list of nearby facilities below the map, each card exposing distance, travel times, facility capabilities, and two action buttons (call and navigate), allowing the user to make an informed choice rather than defaulting blindly to the nearest result.

The rest of this paper is organised as follows. Section II reviews related work. Section III describes the overall system and MVVM architecture. Section IV documents the application's five screens and their ViewModel bindings. Section V details each functional module. Section VI explains the ML inference pipeline. Section VII covers the healthcare proximity and facility-card system. Section VIII describes the responsive layout engine. Section IX discusses testing strategy and architectural testability. Section X addresses limitations and future work, and Section XI concludes.

II. RELATED WORK

A. Mobile Emergency Response and the Golden Hour

The concept that rapid intervention within the first hour of a traumatic event decisively determines patient outcomes is well established in trauma medicine [1]. Mobile applications have been explored as low-cost supplements to formal emergency infrastructure. India's national 112 emergency number consolidates police, fire, and medical dispatch, but it provides no injury-specific guidance and depends entirely on voice communication. Research into mobile-assisted first response has consistently shown that audio and visual prompts significantly improve bystander CPR quality and other first-response actions compared to verbal instruction alone [3]. Help Link extends this principle to a broader range of injuries by combining image-based classification with step-by-step, optionally audio-narrated instructions.

B. On-Device Deep Learning for Wound and Injury Classification

Convolutional neural networks have reached expert-level accuracy in skin-lesion and wound classification tasks [4]. Transfer learning from large-scale architectures such as ResNet [5] and MobileNetV2 [6] dramatically reduces the labelled-data requirement for new domains, making on-device injury classification practical with a modest training corpus. TensorFlow Lite [7] compresses trained models into a flat-buffer format suitable for real-time inference on ARM mobile processors, eliminating cloud round-trip latency and preserving full functionality in offline environments. Help Link loads model_unquant.tflite via the tflite_v2 Flutter plugin with MobileNet-compatible normalisation (pixel mean 127.5, standard deviation 127.5), placing inference entirely on-device.

C. Cross-Platform Development With Flutter

Flutter compiles Dart code to native ARM binaries for iOS and Android from a single codebase, offering performance indistinguishable from platform-native code for the UI-intensive patterns of an emergency application [8]. Its reactive ChangeNotifier model makes it well matched to MVVM: widgets observe ViewModels and rebuild automatically when notifyListeners() is called, without requiring a third-party state-management library. Prior healthcare applications on Flutter report both rapid iteration cycles and consistent performance across low-end and flagship devices [9], both important for a safety-critical application.

D. GPS-Based Healthcare Facility Discovery

Location-based hospital discovery has been demonstrated to improve emergency response efficiency by eliminating the time users spend calling hospitals to confirm availability [10]. The Haversine formula provides an accurate great-circle

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

distance between GPS coordinates for the sub-50-km ranges typical of urban and peri-urban hospital searches, with a maximum error well below 0.5% for the distances involved [11]. Help Link augments raw distance with derived walking (5 km/h) and driving (30 km/h) time estimates, allowing users to make a rapid triage decision between walking to a nearby clinic and driving to a more distant hospital with trauma capability.

E. Text-to-Speech for Emergency Guidance

Audio delivery of first-aid instructions reduces cognitive load and supports compliance during high-stress situations where reading accuracy degrades [12]. It also serves users with visual impairments or low literacy — demographics that are disproportionately represented in India's rural populations. Help Link uses flutter_tts configured at a deliberate 0.45× speech rate, slower than the library default, to ensure that safety-critical negations such as 'do not apply ice' and 'do not burst blisters' are clearly audible in noisy environments.

III. SYSTEM ARCHITECTURE

A. Model-View-ViewModel (MVVM)

Help Link adopts a strict MVVM pattern. The three layers are defined as follows:

- Model layer: Pure Dart data classes — EmergencyContact (name, number), HealthcareLocation (id, name, type, position, address, distanceKm, phone, capabilities, and two computed travel-time getters), and InjuryClassificationResult (label, confidence, minimumConfidence, and three computed Boolean properties: hasPrediction, isConfident, isUnclear). Stateless Service classes encapsulate all business logic: EmergencyContactsService returns the hardcoded contact list; InjuryClassifierService wraps the tflite_v2 plugin; FirstAidService maps injury labels to instruction strings; HealthcareLocationsService implements the Haversine search; LocationService wraps the Geolocator plugin.
- ViewModel layer: Classes extending ChangeNotifier — EmergencyContactsViewModel, InjuryIdentificationViewModel, and GoogleMapViewModel — orchestrate service calls, hold all UI-observable state (loading flags, error messages, classification results, nearby facilities), and expose public methods for every user action. Error states are propagated as nullable String errorMessage fields rather than thrown exceptions, so the View layer can surface them as non-fatal SnackBar messages.
- View layer: Flutter Stateless and Stateful widgets observe ViewModels via AnimatedBuilder or addListener(). No widget calls a Service method directly. Navigation uses MaterialPageRoute; modal dialogs (the first-aid steps dialog) use showDialog().

B. Entry Point and Initialisation

main() calls WidgetsFlutterBinding.ensureInitialized() and locks orientation to portrait-up and portrait-down via SystemChrome.setPreferredOrientations() before running MyApp. The root MaterialApp configures Material Design 3 (useMaterial3: true), a seed colour of 0xFF4E0D80 (deep violet), the Google Fonts Laila text theme, scaffold background 0xFFF6F7FB, and a debug banner suppressed for production presentation. The app launches MySplash, which shows a scale-animated glassmorphic card with a linear progress indicator for 2 400 ms and then navigates to MyHomeScreen via pushReplacement, ensuring no back-stack entry remains on the splash screen.

C. Package Structure

Package	Contents and Responsibility
core/layout/	AppScreenSize enum; screenSizeForWidth(); ResponsiveBuildContext extension (responsive(), pagePadding, maxContentWidth, isCompact/isMedium/isExpanded); ResponsiveContent scroll-and-constrain wrapper.
models/	EmergencyContact, HealthcareLocation (with estimatedWalkMinutes and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

	estimatedDriveMinutes getters), InjuryClassificationResult (with hasPrediction, isConfident, isUnclear, confidencePercent getters).
services/	EmergencyContactsService, InjuryClassifierService (TFLite lifecycle + parseClassificationResult + normalizeLabel), FirstAidService (instructionsFor + displayResultFor), HealthcareLocationsService (Haversine search), LocationService (GPS + permission).
viewmodels/	EmergencyContactsViewModel (direct call), InjuryIdentificationViewModel (model init + image pick + classify + dispose), GoogleMapViewModel (GPS load + nearby search).
screens/	MySplash, MyHomeScreen, EmergencyContactsScreen, InjuryIdentificationScreen (+ _FirstAidStepsDialog), MyGoogleMaps (+ _HealthcareFacilityCard + _InfoChip).

Table I: Package Structure of Help Link

IV. APPLICATION SCREENS AND VIEWMODEL BINDINGS

A. Splash Screen (MySplash)

MySplash is a StatefulWidget whose initState() schedules a 2 400 ms delayed navigation to MyHomeScreen. The UI presents a centred glassmorphic container (white at 86% opacity, rounded corners at 32 dp radius, deep-purple drop shadow) containing the app logo, title, tagline, and an indeterminate LinearProgressIndicator with a 0xFF6A0DAD fill colour. A TweenAnimationBuilder scales the card from 0.92 to 1.0 over 900 ms with an easeOut curve, providing a smooth entrance animation.

B. Home Screen (MyHomeScreen)

MyHomeScreen is a StatelessWidget. Its body wraps a Column inside ResponsiveContent, showing a doctor illustration asset (190 dp tall on compact, scaling to 280 dp on expanded), a bold headline ('Quick help when every second matters'), a subtitle, and a LayoutBuilder-driven Wrap of two _HomeActionCard widgets. On compact screens the cards stack vertically at full width; on medium and expanded screens they sit side-by-side at 50% width each. Tapping Injury Identification pushes InjuryIdentificationScreen; tapping Emergency Contacts pushes EmergencyContactsScreen.

C. Emergency Contacts Screen (EmergencyContactsScreen)

EmergencyContactsScreen is a StatelessWidget that instantiates EmergencyContactsViewModel in-line. An AnimatedBuilder wraps the list so that any ViewModel state change triggers a rebuild. Contacts are displayed as Material ListTile cards in a responsive Wrap (single-column on compact, two-column on medium/expanded). Each tile shows a purple-background phone icon, the service name in Lateef font at 28–32 sp depending on breakpoint, the number in the app's accent purple, and a green call icon button that invokes _viewModel.callContact(). Errors (e.g., device cannot place calls) surface as SnackBars.

D. Injury Identification Screen (InjuryIdentificationScreen)

InjuryIdentificationScreen is a StatefulWidget. initState() instantiates InjuryIdentificationViewModel, registers a listener for error-message changes, and calls _viewModel.initialize() to pre-load the TFLite model asynchronously, displaying 'Preparing the injury model...' in the image placeholder while isPreparingModel is true. The screen body is an AnimatedBuilder observing the ViewModel. When no image is selected, a dashed-border placeholder with an add-photo icon is shown. After image selection, the image is rendered in a ClipRRect at 250–380 dp height. A result card below the image displays displayResult (a plain-English injury description) or a loading spinner while isLoading is true. When classification.isConfident is true, two gradient action buttons appear: 'First Aid Steps' (red-to-orange gradient) and 'Nearest Healthcare' (teal-to-blue gradient). Tapping 'First Aid Steps' calls showDialog() with _FirstAidStepsDialog, which displays the numbered instruction string in a scrollable AlertDialog and includes a flutter_tts toggle button in the title bar. Tapping 'Nearest Healthcare' pushes MyGoogleMaps.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

E. Google Maps Screen (MyGoogleMaps)

MyGoogleMaps is a StatefulWidget. initState() creates GoogleMapViewModel and immediately calls loadCurrentLocation(), which acquires GPS permission, fetches the device position, and runs the Haversine proximity search with a 10 km radius. The screen body is an AnimatedBuilder. Once data is available, a LayoutBuilder allocates 42% of screen height (clamped to 260–380 dp) to a GoogleMap widget and the remaining height to a scrollable list of _HealthcareFacilityCard widgets. The map renders custom-coloured markers — red (BitmapDescriptor.hueRed) for Hospitals, orange (BitmapDescriptor.hueOrange) for all other facility types — with info-window snippets showing facility type and distance. Tapping a list card animates the map camera to the facility at zoom level 16 and then launches turn-by-turn navigation.

V. FUNCTIONAL MODULES

A. Emergency Contact Module

EmergencyContactsService.getContacts() returns a compile-time constant list of 15 EmergencyContact objects covering five categories:

Category	Service Name	Number
Core Emergency	Police	100
Core Emergency	Ambulance	102
Core Emergency	Fire Service	101
Core Emergency	Disaster Management	108
Road & Highway	National Highway Helpline	1033
Road & Highway	Road Accident Emergency	1073
Road & Highway	Motor Vehicle Accident	1099
Medical	Blood Bank	104
Medical	AIIMS Emergency	011-26588500
Safety & Support	Women Helpline	1091
Safety & Support	Child Helpline	1098
Safety & Support	Senior Citizen Helpline	14567
Safety & Support	Mental Health (iCall)	9152987821
Utilities	Cyber Crime Helpline	1930
Utilities	Gas Leak	1906

Table II: Pre-Loaded Emergency Contacts in EmergencyContactsService

Because this list is a Dart const, it requires no file I/O, database query, or network call and is available immediately at app start. EmergencyContactsViewModel.callContact() invokes FlutterPhoneDirectCaller.callNumber(), which triggers a native call intent without presenting the system dialler, reducing the time-to-call to a single tap.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

B. Injury Classification Module

InjuryClassifierService manages the full TFLite lifecycle. ensureModelLoaded() invokes Tflite.loadModel() with the asset path assets/models/model_unquant.tflite and companion labels file assets/models/labels.txt. Key parameters: numThreads = 1 (single-threaded CPU inference for broad device compatibility), useGpuDelegate = false (GPU delegate unavailable on many Android versions). A boolean _isModelLoaded guard prevents redundant re-loading.

classifyImage(File) calls Tflite.runModelOnImage() with imageMean = 127.5 and imageStd = 127.5 — the normalisation that shifts raw [0, 255] pixel values to the [−1, 1] range expected by MobileNet-family architectures — with numResults = 1 (only the top label is needed), threshold = 0 (raw confidence flows to application layer), and asynch = true (inference executes on a background isolate, keeping the UI thread unblocked). The raw result list is passed to the static, unit-testable parseClassificationResult(), which selects the entry with the highest numeric confidence and calls normalizeLabel() to strip any leading integer index (e.g. '0 Burns' → 'Burns') using the regex ^s*d+s+. The cleaned label and confidence are wrapped in an InjuryClassificationResult. dispose() calls Tflite.close() to release native model memory when the screen is popped.

C. First Aid Guidance Module

FirstAidService.instructionsFor() matches injury labels via String.contains() and returns fully formatted, numbered first-aid step strings. Six categories are covered:

Injury Category	Key Guidance Provided by FirstAidService
Bruises	RICE protocol (Rest, Ice 15–20 min wrapped in cloth, Compress, Elevate above heart level); warm compress after 48 h; avoid aspirin (worsens bleeding); medical review if bruise is unexplained, very large, or persists beyond 2 weeks.
Abrasions	Hand hygiene before contact; 5–10 min irrigation with clean running water; mild soap around (not inside) wound; antibiotic ointment; sterile non-stick cover; daily dressing change; watch for infection signs; tetanus note if dirty/rusty object.
Burns	Warning header: do not use ice, butter, toothpaste, or oil. Cool burn under cool (not cold) running water for ≥ 20 min; do not burst blisters; loose cling-film cover; call emergency services if burn > palm size, affects face/hands/feet/joints, is deep/white/charred, caused by chemicals/electricity, or patient is a child or elderly.
Cuts	Firm pressure 5–10 min; irrigate after bleeding stops; antibiotic ointment; adhesive closure for small cuts; go to emergency if bleeding uncontrolled after 10 min, cut is gaping/deep, bone/fat visible, or dirty/rusty cause; tetanus note.
Lacerations	Steady pressure; if object embedded, do not remove — apply pressure around it; elevation above heart; no self-closure of deep/jagged wounds; loose sterile dressing; seek immediate care — stitches, staples, or medical glue often required; tetanus referral.
Puncture Wounds	Do not remove any lodged object; allow brief bleeding to flush bacteria; rinse wound perimeter only (do not probe); clean sterile bandage; prompt medical review essential — deep infection and tetanus risk even from minor-looking punctures; tetanus booster if unsure of vaccination status.

Table III: First Aid Instruction Coverage by Injury Category (FirstAidService)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

A general four-step fallback ('Stay calm and assess the situation. Call for emergency help if needed. Do not move someone if a spinal injury is suspected. Keep the person warm and comfortable until help arrives.') is returned for any label that does not match the six categories above.

The _FirstAidStepsDialog widget presents instructions in a scrollable AlertDialog. The title bar includes a flutter_tts toggle button (volume_up / stop_circle icon). The TTS engine is initialised with language 'en-US', speechRate 0.45, volume 1.0, and pitch 1.0. The deliberately low speech rate ensures that safety-critical negations (e.g., 'Never put ice directly on skin') are intelligible in a noisy emergency environment. A setCompletionHandler and a setCancelHandler both reset the _isSpeaking flag to prevent stale button state after the audio finishes or is interrupted.

VI. ML INFERENCE PIPELINE

A. Model Asset

The classification model is deployed as assets/models/model_unquant.tflite. The 'unquant' suffix indicates that the model retains 32-bit floating-point weights rather than being post-training quantised to INT8. This choice preserves maximum inference accuracy — a priority for a safety-critical application where a misclassified burn could receive cut-oriented first-aid instructions — at the cost of a larger asset size compared to a quantised variant. A companion assets/models/labels.txt file maps output tensor index positions to human-readable class names, with one label per line.

B. Inference Parameter Summary

Parameter	Value and Engineering Rationale
imageMean	127.5 — shifts 0–255 raw pixel values by subtracting 127.5 before division.
imageStd	127.5 — divides shifted values to produce the normalised [−1, 1] input range required by MobileNet-family architectures.
numResults	1 — only the top-ranked label is returned; simplifies downstream parsing in parseClassificationResult().
threshold	0 — no plugin-level filtering; the application enforces its own 0.80 gate in InjuryClassificationResult.isConfident.
asynch	true — TFLite inference runs on a background isolate, ensuring the Flutter UI thread remains unblocked during processing.
numThreads	1 — single-threaded inference maximises compatibility across low-RAM and entry-level Android devices.
useGpuDelegate	false — CPU inference used for guaranteed compatibility; the Android GPU delegate is unavailable on API levels below 27 and on certain chipsets.

Table IV: TFLite Inference Configuration Parameters

C. Confidence Gating and Threshold Rationale

InjuryClassificationResult defines a minimum confidence of 0.80. Results below this threshold set isConfident to false and isUnclear to true (when hasPrediction is true but confidence is insufficient). The application maps these states to distinct UI messages: 'No visible injury found.' is shown when an image is present but confidence is below threshold, prompting the user to re-photograph under better conditions. This gating prevents first-aid instructions for the wrong injury from being presented — a safety risk more serious than showing no result.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

D. Label Normalisation

Raw TFLite output labels may carry a leading zero-indexed integer prefix generated during model training (e.g., '0 Burns', '3 Cuts'). InjuryClassifierService.normalizeLabel() removes this prefix with the regular expression ^s*d+s+ applied via String.replaceFirst(). FirstAidService then matches normalised labels (e.g., 'Burns', 'Cuts', 'Laceration') with String.contains() rather than exact equality, providing tolerance for minor label-text variations between model versions without requiring FirstAidService changes.

VII. HEALTHCARE PROXIMITY AND FACILITY CARD SYSTEM

A. Haversine Distance Engine

HealthcareLocationsService._calculateDistance() implements the Haversine formula using dart:math. Given two LatLng coordinates, it computes the central angle using the formula $a = \sin^2(\Delta lat/2) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2(\Delta lng/2)$ and the great-circle distance as $d = 2 \cdot R \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$, where R = 6 371 km is the mean Earth radius. This formula provides accurate distances for the sub-50-km ranges relevant to urban hospital searches. findNearbyHealthcare() filters the full facility list to results within the search radius (10 km as called from GoogleMapViewModel.loadCurrentLocation()) and sorts by ascending distance. Each HealthcareLocation exposes two computed travel-time getters: estimatedWalkMinutes = ceil(distanceKm / 5.0 × 60) and estimatedDriveMinutes = ceil(distanceKm / 30.0 × 60), using the ceiling function so that a 0.1 km walk does not round down to zero minutes.

B. Facility Card Design (_HealthcareFacilityCard)

Each facility is rendered as a Material card with rounded corners (radius 14 dp) and a white fill at 92% opacity. The card is structured as a horizontal Row with three zones:

- Left accent strip (60 dp wide): Displays a type-appropriate icon (Icons.local_hospital for Hospital, Icons.emergency for Urgent Care, Icons.medical_services for Clinic/other) on a solid background colour that is determined by the _accentColor getter — red (Colors.red.shade600) for Hospitals, orange (Colors.orange.shade700) for Urgent Care, teal for all others. This colour is reused throughout the card for semantic consistency.
- Centre detail column: Shows the facility name in bold Laila font (14 sp), a pill-shaped type badge (e.g., 'Hospital', 'Urgent Care') in the accent colour at reduced opacity, the street address with a location_on icon, a Wrap row of three _InfoChip widgets — distance in km, estimated walk time in minutes, estimated drive time in minutes — and a second Wrap of capability chips. Chips for 'Emergency' and '24hr' labels are rendered with a tinted background and bold text in the accent colour; all other capability chips (e.g., 'ICU', 'Surgery', 'Pharmacy') use a neutral grey.
- Right action column: Two IconButton widgets — a green phone icon that calls FlutterPhoneDirectCaller.callNumber(location.phone) and a coloured navigation icon that calls _startNavigation(), which attempts to launch the google.navigation: deep-link URI first and falls back to a google.com/maps/dir/ web URL if the Google Maps application is not installed.

The _startNavigation() method also animates the GoogleMap camera to the selected facility at zoom level 16 before launching the external navigation intent, giving the user a visual confirmation of which facility they are navigating to. Map markers are colour-coded identically to the card accent colours (red hue for Hospitals, orange hue for all others) through BitmapDescriptor.defaultMarkerWithHue(). InfoWindow snippets display the facility name as the title and 'Type • X.X km' as the snippet, providing a quick reference without opening the full list card.

VIII. RESPONSIVE LAYOUT SYSTEM

Help Link implements a bespoke responsive layout system in core/layout/responsive.dart rather than depending on third-party adaptive packages. The system defines three breakpoints via the AppScreenSize enum — compact (< 600

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

dp width), medium (600–1023 dp), expanded (⩾ 1024 dp) — aligned with the Material Design 3 canonical layout specification.

The ResponsiveBuildContext extension on BuildContext exposes five members:

- responsive({compact, medium?, expanded?}): Returns the value for the current breakpoint, falling back to the next smaller value when an override is absent. Used for font sizes, padding amounts, card heights, and icon dimensions throughout every screen.
- pagePadding: Returns symmetric EdgeInsets with horizontal padding 16/24/32 dp and vertical padding 16/20/24 dp for compact/medium/expanded respectively.
- maxContentWidth: Constrains content to 560/760/1040 dp for compact/medium/expanded, preventing excessively long text lines on large tablets and foldables.
- isCompact / isMedium / isExpanded: Boolean accessors used for conditional layout decisions, such as switching a Wrap from a full-width single-column arrangement to a two-column arrangement on medium and expanded screens.

ResponsiveContent wraps all screen bodies. It uses LayoutBuilder inside SafeArea to centre content, apply pagePadding, constrain width to maxContentWidth, and optionally wrap in a SingleChildScrollView (scrollable: true by default). The scrollable wrapper uses a ConstrainedBox with minHeight equal to the viewport height, ensuring that full-screen-height content (such as the splash card) remains vertically centred even when the child is short.

IX. TESTING AND ARCHITECTURAL TESTABILITY

A. Unit-Testable Static Methods

InjuryClassifierService exposes two static methods annotated @visibleForTesting that are designed exclusively for unit testing. parseClassificationResult(List<dynamic>?, {double minConfidence}) accepts a raw TFLite result list and returns a fully typed InjuryClassificationResult without requiring a device, emulator, or loaded model. extractAcceptedLabel() wraps parseClassificationResult() and returns the label string only when isConfident is true, or an empty string otherwise. These methods allow deterministic tests to verify: null/empty input handling, confidence threshold enforcement at the boundary (exactly 0.80), the hasPrediction/isConfident/isUnclear state machine, label normalisation via normalizeLabel(), and the _readConfidence() coercion of heterogeneous numeric types.

B. Dependency Injection in ViewModels

All three ViewModels accept their service dependencies as optional constructor parameters with default values (e.g., InjuryIdentificationViewModel({InjuryClassifierService? classifierService, FirstAidService? firstAidService, ImagePicker? imagePicker})). This constructor-injection pattern permits test doubles to be substituted for all services in unit tests, enabling verification of ViewModel state transitions (isLoading, errorMessage, classification, isPreparingModel) and error-propagation paths without any Flutter rendering, live GPS signal, camera access, or TFLite model.

C. Error Propagation Strategy

All async ViewModel methods follow a consistent try-catch-finally pattern. The finally block always clears the loading flag and calls notifyListeners(), ensuring the UI never becomes stuck in a loading state even after an unexpected exception. Errors are assigned to errorMessage (a nullable String field) rather than re-thrown, so they propagate to the View as non-fatal SnackBar messages rather than unhandled exceptions that would crash the application. LocationPermissionException is a typed custom exception thrown by LocationService to distinguish the permission-denied case from other GPS errors, allowing GoogleMapViewModel to surface a specific, actionable error message to the user.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

X. LIMITATIONS AND FUTURE WORK

A. Current Limitations

The healthcare facility dataset used by HealthcareLocationsService is currently populated with placeholder data (mock coordinates offset from the user's GPS position). Production deployment requires integration with a live, verified healthcare directory — such as the National Health Authority's Ayushman Bharat Digital Mission Health Facility Registry — or periodic synchronisation to a local SQLite database. The first-aid service covers six injury categories; medically important categories such as fractures, sprains, head trauma, choking, and anaphylaxis are absent and fall through to a generic four-step fallback. The training dataset size, class distribution, and held-out accuracy metrics for model_unquant.tflite are not documented in the codebase; for clinical credibility, these should be published alongside the application. The unquantised model carries a larger asset bundle size than an INT8-quantised equivalent, which may affect cold-start time on devices with limited flash storage.

B. Future Work

Five enhancements are prioritised for future development iterations:

1. Live healthcare directory integration via the NHA Health Facility Registry REST API, with periodic background synchronisation to a local SQLite cache using the sqflite Flutter plugin.
2. Expansion of both the TFLite model and FirstAidService to cover fractures, sprains, head injuries, anaphylaxis, and choking — the injury categories most prevalent in Indian road-accident and household emergency data.
3. Multi-language localisation using Flutter's built-in intl package, prioritising Hindi and four major regional languages (Tamil, Telugu, Kannada, Marathi), with corresponding TTS language codes passed to flutter_tts.
4. INT8 post-training quantisation of model_unquant.tflite using TensorFlow Lite's quantisation API to reduce bundle size and improve inference latency on low-RAM devices, with accuracy benchmarking to verify that the 80% confidence threshold remains appropriate post-quantisation.
5. Automatic emergency SMS dispatch: when a user dials an emergency number, optionally send an SMS containing the device's GPS coordinates and the classified injury label to a pre-registered emergency contact, using the telephony Flutter plugin.

XI. CONCLUSION

This paper presented Help Link, a Flutter mobile emergency assistance application that integrates on-device TFLite injury classification, a 15-contact one-tap emergency dialling module, Haversine-based nearest-healthcare discovery with rich facility cards, and TTS-narrated first-aid instructions into a single offline-capable system. The application's strict MVVM architecture — with constructor-injected services, static unit-testable parsing methods, and consistent try-catch-finally error propagation — provides a maintainable foundation that separates UI, business logic, and data access cleanly across three well-defined layers.

The healthcare discovery screen represents the most feature-rich component of the application: each facility card surfaces facility type, a colour-coded accent, street address, capabilities, Haversine-computed distance, walking and driving time estimates, a direct-call button, and a turn-by-turn navigation launcher — giving users sufficient information to triage between nearby facilities rather than defaulting blindly to the geographically nearest result. Combined with the 80%-gated injury classifier and step-by-step first-aid instructions narrated at a deliberately slow TTS rate, Help Link provides a comprehensive, accessible, and offline-capable emergency assistance experience that requires no internet connectivity for its four core functions.

Help Link demonstrates that production-grade AI-assisted emergency assistance is achievable on a mobile device with no cloud dependency, making it viable in rural and peri-urban Indian contexts where emergency infrastructure and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

connectivity are most limited and where rapid, informed first response has the greatest potential to improve survival outcomes.

REFERENCES

[1] American College of Surgeons Committee on Trauma, Advanced Trauma Life Support (ATLS) Student Course Manual, 10th ed. Chicago, IL: American College of Surgeons, 2018.
[2] Ministry of Road Transport and Highways, Government of India, Road Accidents in India 2022. New Delhi: MoRTH, 2023.
[3] T. Iwami, T. Kitamura, T. Kawamura, and A. Hiraide, "Chest compression-only cardiopulmonary resuscitation for out-of-hospital cardiac arrest with public-access defibrillation: A nationwide cohort study," Circulation, vol. 126, no. 24, pp. 2844–2851, Dec. 2012.
[4] A. Esteva et al., "Dermatologist-level classification of skin cancer with deep neural networks," Nature, vol. 542, pp. 115–118, Feb. 2017.
[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520.
[7] R. David et al., "TensorFlow Lite: On-device machine learning for mobile and edge computing," arXiv:2010.14335, Oct. 2020.
[8] Google LLC, Flutter — Build apps for any screen. [Online]. Available: https://flutter.dev. [Accessed: Apr. 2026].
[9] S. R. Steinhubl, E. D. Muse, and E. J. Topol, "Can mobile health technologies transform health care?," JAMA, vol. 320, no. 23, pp. 2395–2396, Dec. 2018.
[10] P. Ageron, C. Gauss, and P. Ravaud, "Ambulance dispatch by Global Positioning System and hospital routing: A systematic review," Emergency Medicine Journal, vol. 29, no. 3, pp. 175–181, Mar. 2012.
[11] C. H. D. Villanueva, "Solving the distance problem: Spherical versus Haversine models for sub-50-km geodesic calculations," Journal of Geodesy, vol. 87, no. 4, pp. 355–362, Apr. 2013.
[12] J. A. Soar, J. P. Nolan, and R. W. Koster, "European Resuscitation Council guidelines for resuscitation 2015: Section 3 — Adult advanced life support," Resuscitation, vol. 95, pp. 100–147, Oct. 2015.
[13] G. Litjens et al., "A survey on deep learning in medical image analysis," Medical Image Analysis, vol. 42, pp. 60–88, Dec. 2017.
[14] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.