Palli Sahayak: A World-Class Voice AI Agent Helpline for Democratizing Palliative Care

A Digital Public Good for Global Health Equity

Version 1.0 | December 2025

Executive Summary

Palli Sahayak has evolved from a text-based RAG (Retrieval-Augmented Generation) system into a comprehensive **Voice AI Agent Helpline** designed to democratize palliative care knowledge across India and the developing world. This document presents the enhanced architecture incorporating state-of-the-art (SOTA) technologies including:

- **Bolna.ai Voice AI Integration** for real-time phone call support
- **Google Gemini Live API** for web-based voice conversations
- **Knowledge Graph with Neo4j** for enhanced medical entity relationships
- **Hybrid RAG Architecture** with intelligent context fusion
- **Multilingual Support** for 15+ Indian languages

By positioning Palli Sahayak as a **Digital Public Good (DPG)**, we aim to address the critical gap in palliative care access affecting over 10 million patients annually in India alone, while providing a replicable model for low- and middle-income countries (LMICs) worldwide.

Table of Contents

---

1. Introduction and Global Context

1.1 The Palliative Care Crisis

Palliative care, defined by the WHO as "an approach that improves the quality of life of patients and their families facing problems associated with life-threatening illness," remains critically underserved globally:

- **India**: Only 1-2% of the 10+ million patients needing palliative care have access to it
- **Global**: 57 million people require palliative care annually; 78% live in LMICs
- **Healthcare Worker Shortage**: WHO projects a shortfall of 10 million health workers by 2030, predominantly in LMICs

1.2 The Opportunity

India's unique digital infrastructure presents an unprecedented opportunity:

- **500+ million** WhatsApp users
- **1.2 billion** mobile phone subscribers
- **IndiaAI Mission** investing significantly in foundational AI models
- **Bhashini** initiative supporting 22 scheduled languages
- Growing telemedicine adoption post-COVID-19

1.3 Palli Sahayak's Mission

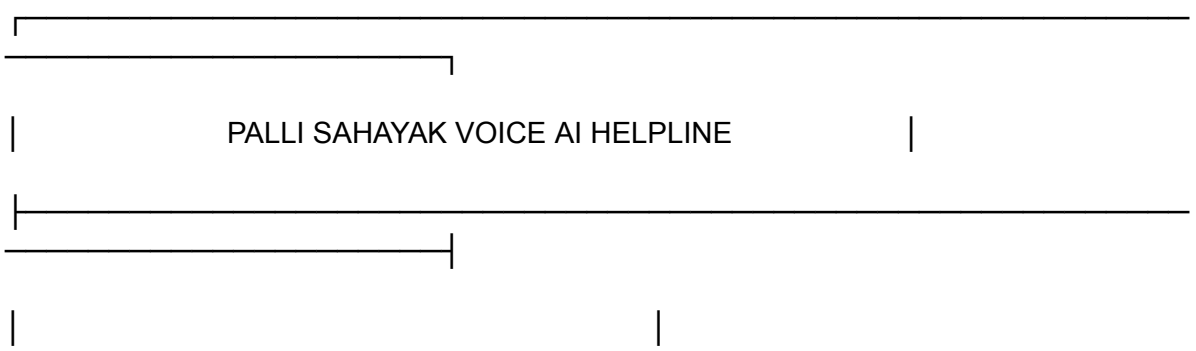
Palli Sahayak ("Companion in Care") aims to:

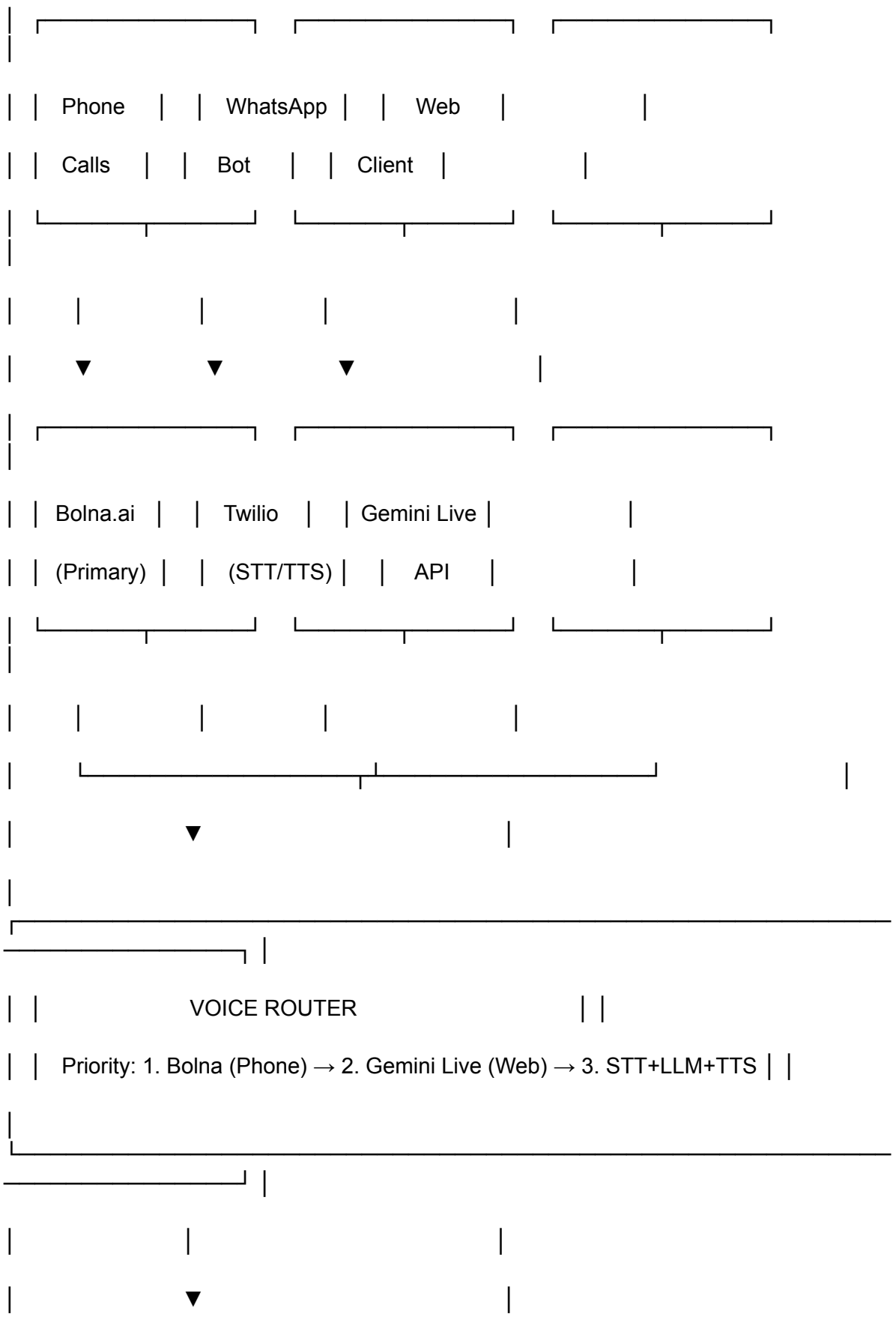
1. **Democratize** access to evidence-based palliative care information
2. **Empower** community health workers (ASHA, ANM) with clinical decision support
3. **Bridge** linguistic and technological barriers
4. **Scale** quality palliative care knowledge to millions

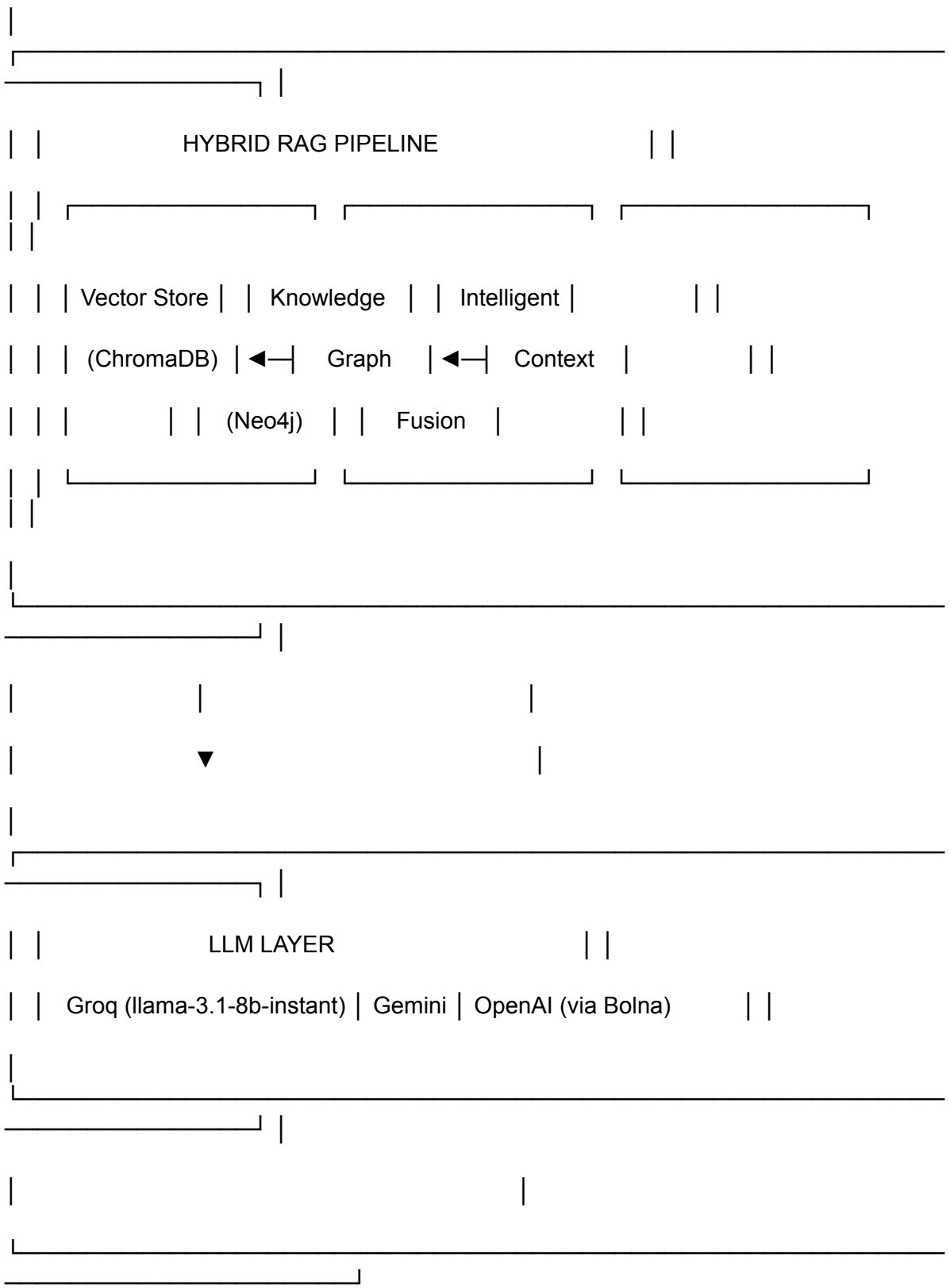
---

2. System Architecture Overview

2.1 High-Level Architecture




Phone Calls
WhatsApp Bot
Web Client
Bolna.ai (Primary)
Twilio (STT/TTS)
Gemini Live API
VOICE ROUTER
Priority: 1. Bolna (Phone) → 2. Gemini Live (Web) → 3. STT+LLM+TTS


HYBRID RAG PIPELINE
Vector Store
(ChromaDB)
Knowledge
Graph
(Neo4j)
Intelligent
Context
Fusion
LLM LAYER
Groq (llama-3.1-8b-instant)
Gemini
OpenAI (via Bolna)

2.2 Core Components

Component	Technology	Purpose
FastAPI Server	Python 3.11+	API backbone, webhook handling
RAG Pipeline	Kotaemon-based	Document processing, retrieval, generation
Vector Store	ChromaDB	Semantic similarity search
Knowledge Graph	Neo4j	Medical entity relationships
Voice AI (Phone)	Bolna.ai	Real-time phone conversations
Voice AI (Web)	Gemini Live API	WebSocket-based voice streaming
WhatsApp Bot	Twilio API	Text and voice message handling
STT	Groq Whisper	Speech-to-text transcription
TTS	Edge TTS	Text-to-speech synthesis
Admin UI	Gradio	Web-based administration

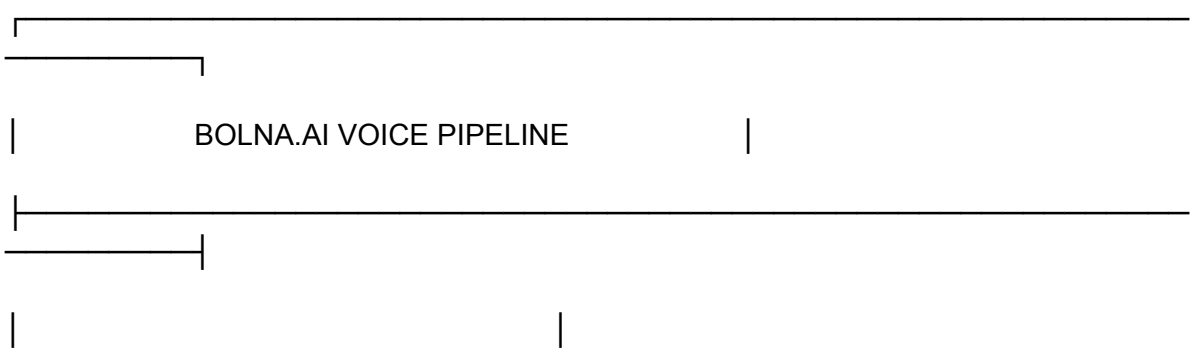
---

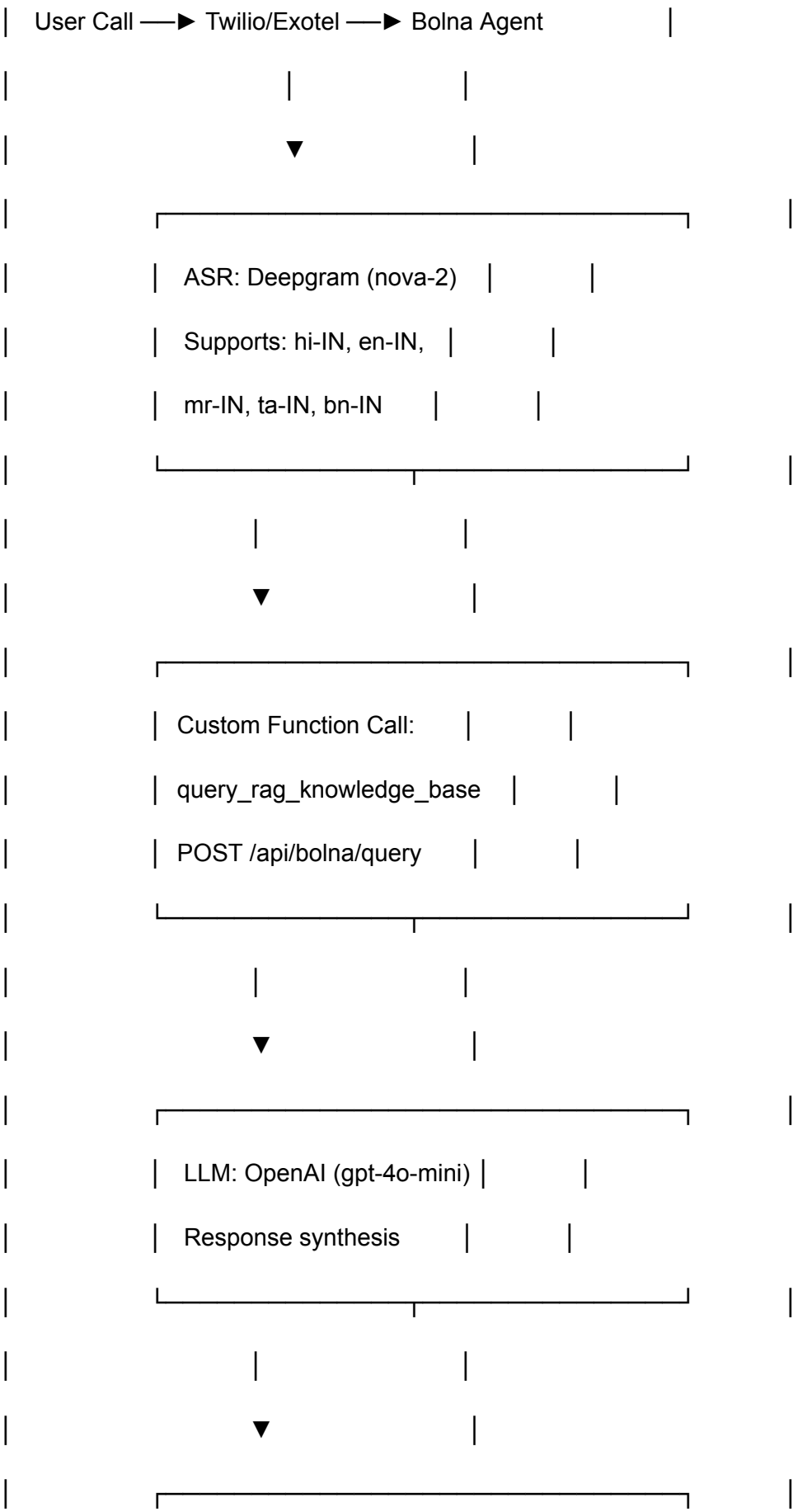
3. Voice AI Integration

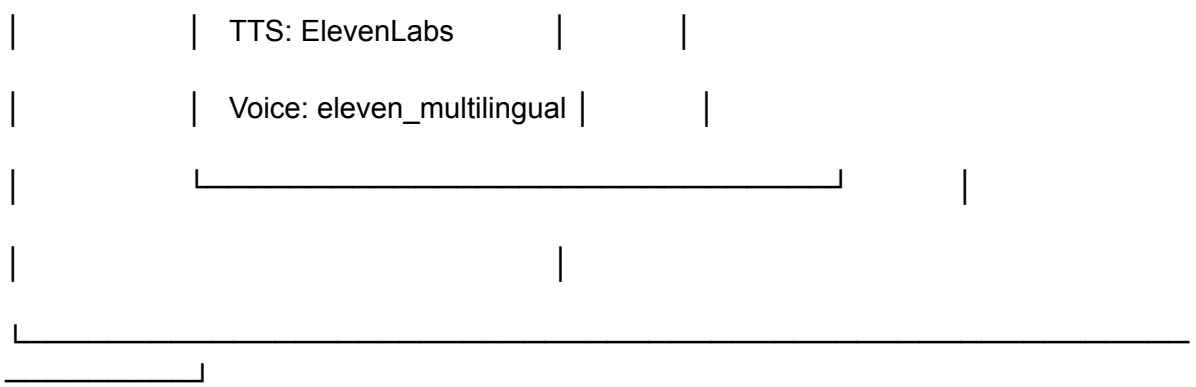
3.1 Bolna.ai Integration (Primary Voice Channel)

Bolna.ai serves as the primary orchestrator for phone-based voice interactions, providing a toll-free helpline experience.

3.1.1 Architecture




User Call ──► Twilio/Exotel ──► Bolna Agent
ASR: Deepgram (nova-2)
Supports: hi-IN, en-IN,
mr-IN, ta-IN, bn-IN
Custom Function Call:
query_rag_knowledge_base
POST /api/bolna/query
LLM: OpenAI (gpt-4o-mini)
Response synthesis



3.1.2 Key Features

- **Real-time Streaming**: Sub-200ms latency for natural conversations
- **Interruption Handling**: Users can interrupt mid-response
- **Context Persistence**: Maintains conversation history within sessions
- **Custom Function Calls**: RAG queries via HTTP POST
- **Webhook Events**: Call start, end, transcript logging

3.1.3 RAG Integration Endpoint

```
@app.post("/api/bolna/query")

async def bolna_query(request: BolnaQueryRequest):

    """

    Endpoint for Bolna custom function calls.

    Receives user query, returns RAG-grounded response.

    """

    result = await rag_pipeline.query(

        question=request.query,

        language=request.language,

        conversation_history=request.history

    )

    return {

        "response": result["answer"],

        "sources": result["sources"],
```

```
        "confidence": result["confidence"]
    }
```

3.2 Google Gemini Live API (Web Voice Channel)

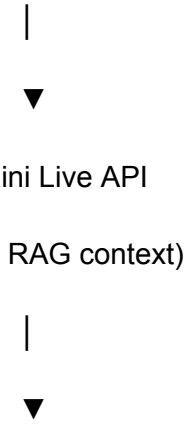
For web-based voice interactions, Gemini Live API provides real-time multimodal conversations.

3.2.1 Technical Specifications

Parameter	**Value**
Input Audio	16-bit PCM, 16kHz, mono, little-endian
Output Audio	16-bit PCM, 24kHz, mono, little-endian
Connection	WebSocket (WSS)
Max Session	15 minutes (audio-only)
Supported Languages	en-IN, hi-IN, mr-IN, ta-IN

3.2.2 Audio Flow

User speaks → Browser AudioWorklet → PCM (16kHz) → WebSocket



User hears ← Web Audio API ← PCM (24kHz) ← WebSocket response

3.2.3 RAG Context Injection

```
class GeminiLiveService:
    async def inject_rag_context(self, session, query: str):
        """
        Inject relevant RAG context into Gemini session.
```

```
    """
    # Query vector DB with recent conversation context
    relevant_chunks = await self.vector_store.similarity_search(
        query, k=5
    )
    # Query knowledge graph for entity relationships
    kg_context = await self.knowledge_graph.get_related_entities(
        entities=extract_entities(query)
    )
    # Format as system instruction
    context = self._format_context(relevant_chunks, kg_context)
    # Send to Gemini as client content
    await session.send({
        "client_content": {
            "turns": [{"role": "user", "parts": [{"text": context}]}],
            "turn_complete": True
        }
    })
```

3.3 Voice Router and Fallback Strategy

The `VoiceRouter` intelligently routes voice interactions:

```
class VoiceRouter:
    """
    Priority-based routing for voice interactions.
    Priority Order:
```

```
1. Bolna.ai - Phone calls (most robust)

2. Gemini Live - Web voice (real-time streaming)

3. STT + RAG + TTS - Ultimate fallback (highest compatibility)

"""

async def route(self, request: VoiceRequest) -> VoiceResponse:

    if request.channel == "phone":

        return await self.bolna_handler(request)

    elif request.channel == "web" and self.gemini_available:

        return await self.gemini_handler(request)

    else:

        return await self.fallback_pipeline(request)
```

---

4. Knowledge Graph for Palliative Care

4.1 Overview

The Knowledge Graph module enhances RAG with structured medical entity relationships, inspired by OncoGraph and biomedical NLP best practices.

4.2 Entity Types

The system extracts 18 specialized palliative care entity types:

Entity Type	**Examples**	**Color Code**
Symptom	Pain, nausea, dyspnea, fatigue	#FF6B6B (Red)
Medication	Morphine, ondansetron, haloperidol	#4ECDC4 (Teal)
Condition	Cancer, COPD, heart failure, dementia	#45B7D1 (Blue)

Entity Type	Examples	Color Code
Treatment	Chemotherapy, palliative sedation	#96CEB4 (Green)
SideEffect	Constipation, sedation, dry mouth	#FFEAA7 (Yellow)
CareSetting	Hospice, home care, hospital	#DDA0DD (Plum)
CareGoal	Comfort care, symptom control	#98D8C8 (Mint)
AssessmentTool	VAS, ECOG, PPS, ESAS	#F7DC6F (Gold)
Route	Oral, subcutaneous, IV, transdermal	#BB8FCE (Lavender)
Psychosocial	Grief, caregiver burden, spiritual distress	#85C1E9 (Sky)
AdvanceCarePlanning	DNR, advance directive, living will	#F1948A (Coral)
Intervention	Massage therapy, wound care	#82E0AA (Emerald)

4.3 Relationship Types

(Medication)-[:TREATS]->(Symptom)

(Condition)-[:CAUSES]->(Symptom)

(SideEffect)-[:SIDE_EFFECT_OF]->(Medication)

(Treatment)-[:MANAGES]->(Condition)

(Intervention)-[:ALLEVIATES]->(Symptom)

(AssessmentTool)-[:MEASURES]->(Symptom)

(Route)-[:ADMINISTERS]->(Medication)

4.4 Base Knowledge Graph

The system includes curated palliative care knowledge:

4.4.1 WHO Pain Ladder Relationships

Strong Opioids: morphine, fentanyl, oxycodone, hydromorphone, methadone

└── TREATS: severe pain, cancer pain, breakthrough pain

Weak Opioids: tramadol, codeine

└── TREATS: moderate pain

Non-Opioids: paracetamol, NSAIDs

└── TREATS: mild pain

4.4.2 Symptom Management Protocols

Nausea/Vomiting:

ondansetron, metoclopramide, haloperidol → TREATS → nausea, vomiting

Respiratory:

morphine (low dose), oxygen, fan therapy → TREATS → dyspnea

Anxiety/Agitation:

lorazepam, midazolam, haloperidol → TREATS → anxiety, agitation, delirium

Constipation:

senna, lactulose, bisacodyl, methylnaltrexone → TREATS → constipation

4.5 Hybrid RAG with Knowledge Graph

```
class HybridRAGPipeline:
    """
    Combines vector search with knowledge graph traversal.
    """
    async def query(self, question: str) -> Dict[str, Any]:
        # 1. Extract entities from question
        entities = await self.entity_extractor.extract(question)
        # 2. Vector similarity search
```

```
vector_results = await self.vector_store.search(question, k=5)

# 3. Knowledge graph traversal

kg_results = await self.knowledge_graph.query_entities(entities)

# 4. Merge and rerank results

merged_context = self._merge_contexts(

    vector_results,

    kg_results,

    strategy="reciprocal_rank_fusion"

)

# 5. Generate answer with LLM

answer = await self.llm.generate(

    question=question,

    context=merged_context,

    kg_entities=entities

)

return {

    "answer": answer,

    "sources": self._extract_sources(merged_context),

    "entities": entities,

    "visualization": kg_results.get("visualization")

}
```

4.6 API Endpoints

Endpoint	Method	Description
`/api/kg/health`	GET	Health status check
`/api/kg/stats`	GET	Graph statistics
`/api/kg/query`	POST	Natural language query
`/api/kg/extract`	POST	Entity extraction from text
`/api/kg/entity/{name}`	GET	Get entity subgraph
`/api/kg/treatments/{symptom}`	GET	Find treatments for symptom
`/api/kg/side-effects/{medication}`	GET	Get medication side effects
`/api/kg/visualization/{name}`	GET	Interactive Cytoscape.js visualization

---

5. State-of-the-Art RAG Enhancements

5.1 Current SOTA Techniques (2025)

Based on comprehensive research of RAG architectures, the following SOTA techniques are implemented or planned:

5.1.1 Hybrid Search (Implemented)

Combines dense embeddings with sparse retrieval:

```
class HybridSearcher:
    """
    Combines semantic (dense) and lexical (sparse) search.
    """
    def search(self, query: str, k: int = 5) -> List[Document]:
        # Dense semantic search
```

```
    dense_results = self.embedding_search(query, k=k*2)
    # Sparse BM25 search
    sparse_results = self.bm25_search(query, k=k*2)
    # Reciprocal Rank Fusion
    fused = self.reciprocal_rank_fusion(
        dense_results,
        sparse_results,
        k=60  # RRF parameter
    )
    return fused[:k]
```

5.1.2 Intelligent Context Fusion (Implemented)

Analyzes semantic similarity distances to determine fusion strategy:

```
def intelligent_context_fusion(results: List[Result], threshold: float = 0.15):
    """
    Fuses multiple highly-relevant contexts or selects best single context.
    """
    distances = [r.distance for r in results]
    distance_range = max(distances) - min(distances)
    if distance_range <= threshold:
        # Multiple contexts are similarly relevant - fuse them
        return fuse_contexts(results), {"fusion": True}
    else:
        # One context is clearly best - use it alone
        return [results[0]], {"fusion": False}
```

5.1.3 Reranking with Cross-Encoders (Planned)

```
class CrossEncoderReranker:
    """
    Reranks initial retrieval results using cross-encoder model.
    """
    def __init__(self, model_name: str = "cross-encoder/ms-marco-MiniLM-L-6-v2"):
        self.model = CrossEncoder(model_name)
    def rerank(self, query: str, documents: List[Document], top_k: int = 5):
        pairs = [(query, doc.content) for doc in documents]
        scores = self.model.predict(pairs)
        ranked = sorted(
            zip(documents, scores),
            key=lambda x: x[1],
            reverse=True
        )
        return [doc for doc, _ in ranked[:top_k]]
```

5.1.4 Multi-Hop Reasoning (Planned)

For complex medical queries requiring reasoning across multiple documents:

```
class MultiHopReasoner:
    """
    Decomposes complex queries into sub-questions.
    """
    async def reason(self, query: str) -> Dict[str, Any]:
        # Decompose query
        sub_questions = await self.decompose(query)
```

```
        # Answer each sub-question
        sub_answers = []
        for sq in sub_questions:
            answer = await self.rag_pipeline.query(sq)
            sub_answers.append(answer)
        # Synthesize final answer
        final_answer = await self.synthesize(
            original_query=query,
            sub_answers=sub_answers
        )
        return final_answer
```

5.2 GraphRAG Integration (Planned)

Following Microsoft's GraphRAG approach:

1. **Entity Knowledge Graph Extraction**: LLM extracts entities and relationships from corpus
2. **Community Detection**: Hierarchical clustering of related entities
3. **Community Summaries**: Pre-generated summaries for entity communities
4. **Global Query Handling**: For sensemaking questions, aggregate community summaries

5.3 Confidence-Calibrated RAG (Planned)

```
class CalibratedRAG:
    """
    Provides confidence scores with responses.
    """
    async def query_with_confidence(self, question: str):
        result = await self.rag_pipeline.query(question)
```

```
    # Calculate confidence based on:
    # 1. Retrieval similarity scores
    # 2. Number of corroborating sources
    # 3. LLM self-assessment
    # 4. Knowledge graph support
    confidence = self.calculate_confidence(
        retrieval_scores=result["distances"],
        source_count=len(result["sources"]),
        kg_support=result.get("kg_entities", [])
    )
    return {
        **result,
        "confidence": confidence,
        "confidence_explanation": self.explain_confidence(confidence)
    }
```

---

6. Multilingual Support and Accessibility

6.1 Language Coverage

Language	**Code**	**STT**	**TTS**	**Voice (Neural)**
Hindi	hi-IN	Whisper	Edge TTS	hi-IN-SwaraNeural
English (India)	en-IN	Whisper	Edge TTS	en-IN-NeerjaNeural
Bengali	bn-IN	Whisper	Edge TTS	bn-IN-TanishaaNeural

Language	Code	STT	TTS	Voice (Neural)
Tamil	ta-IN	Whisper	Edge TTS	ta-IN-PallaviNeural
Gujarati	gu-IN	Whisper	Edge TTS	gu-IN-DhwaniNeural
Marathi	mr-IN	Whisper	Edge TTS	mr-IN-AarohiNeural
Telugu	te-IN	Planned	Planned	te-IN-ShrutiNeural
Kannada	kn-IN	Planned	Planned	kn-IN-SapnaNeural

6.2 Cross-Lingual RAG

The system handles queries in local languages against English medical documents:

SYSTEM_PROMPT = """

You are a palliative care assistant. The user may ask in Hindi, Bengali,

Tamil, or other Indian languages. The medical context is in English.

Instructions:

1. Understand the user's question regardless of language

2. Extract relevant information from the English context

3. Respond in the SAME language as the user's question

4. Use simple, clear language appropriate for patients and caregivers

5. Always cite sources when providing medical information

"""

6.3 Accessibility Features

1. **Voice-First Design**: Users can interact entirely through voice
2. **Simple Commands**: `/lang hi` to change language preference
3. **Audio Responses**: All text responses have audio equivalents
4. **Low-Bandwidth Mode**: Compressed audio for rural connectivity
5. **USSD Fallback**: Planned for feature phones

---

7. Digital Public Good Positioning

7.1 What is a Digital Public Good?

Digital Public Goods (DPGs) are open-source software, open data, open AI models, open standards, and open content that adhere to privacy and applicable best practices, do no harm, and help attain the Sustainable Development Goals (SDGs).

7.2 Alignment with DPG Criteria

Criterion	**Palli Sahayak Compliance**
Open Source	MIT License, publicly available on GitHub
SDG Relevance	SDG 3 (Good Health), SDG 10 (Reduced Inequalities)
Privacy	No personal health data stored; anonymized interactions
Do No Harm	Medical guardrails; referral to professionals
Data Standard	Standard medical ontologies (SNOMED, ICD-10)
Interoperability	REST APIs, webhook support, modular architecture

7.3 Global Impact Potential

7.3.1 India Impact

- **Target Users**: 10+ million patients, 1+ million ASHAs/ANMs
- **Languages**: 15+ scheduled languages
- **Reach**: Rural and urban areas via WhatsApp and phone

7.3.2 Global Replicability

The architecture is designed for adaptation:

config.yaml - Example for adaptation to another country

country: "kenya"

languages:

```
  - code: "sw"
    name: "Swahili"
    stt_model: "whisper-large-v3"
    tts_voice: "sw-KE-ZuriNeural"
  - code: "en"
    name: "English"
    tts_voice: "en-KE-AsiliaNeural"
corpus:
  source: "kenya_palliative_care_guidelines"
  format: "pdf"
voice_provider:
  primary: "bolna"  # or "vonage", "twilio"
  fallback: "stt_llm_tts"
```

7.3.3 Partnership Opportunities

1. **WHO**: Global palliative care guidelines dissemination
2. **UNICEF**: Child palliative care in LMICs
3. **Open Source Pharma Foundation**: Medication information
4. **Local Health Ministries**: National guideline integration

7.4 Sustainable Development Goals Impact

SDG	**Contribution**
SDG 3.8	Universal health coverage, access to essential medicines
SDG 3.4	Reduce premature mortality from non-communicable diseases
SDG 10.2	Empower and promote social inclusion
SDG 17.6	Technology transfer and capacity building

---

8. Ethical Considerations and Safety

8.1 Responsible AI Principles

1. **Transparency**: Clear disclosure of AI nature
2. **Accuracy**: Grounded responses from verified medical sources
3. **Privacy**: No personal health information stored
4. **Accountability**: Human oversight and feedback mechanisms
5. **Fairness**: Equal access regardless of language or region

8.2 Medical Safety Guardrails

```
class MedicalSafetyGuard:

    """

    Ensures safe medical information delivery.

    """

    EMERGENCY_KEYWORDS = [

        "bleeding", "unconscious", "not breathing", "chest pain",

        "stroke", "seizure", "suicide", "overdose"

    ]

    DISCLAIMER = """

    This information is for educational purposes only and should not

    replace professional medical advice. Please consult a healthcare

    provider for personalized recommendations.

    """

    async def process(self, query: str, response: str) -> str:

        # Check for emergency situations

        if self.is_emergency(query):

            return self.emergency_response()

        # Add appropriate disclaimers
```

```
    response = self.add_disclaimer(response)

    # Suggest professional consultation for complex cases

    if self.needs_professional(query, response):

        response += "\n\nPlease consult a palliative care specialist."

    return response
```

8.3 Bias Mitigation

1. **Corpus Diversity**: Include guidelines from multiple regions
2. **Language Parity**: Equal quality across all supported languages
3. **Accessibility Testing**: Regular testing with diverse user groups
4. **Feedback Loop**: Continuous improvement based on user feedback

8.4 Data Privacy

- No personal health information (PHI) stored
- Conversation logs anonymized
- Session data expires after 24 hours
- HIPAA-aligned security practices
- Local data sovereignty compliance

---

9. Future Work and Roadmap

9.1 Short-Term (Q1-Q2 2026)

1. **GraphRAG Implementation**: Full Microsoft GraphRAG integration
2. **Cross-Encoder Reranking**: Improved retrieval precision
3. **Additional Languages**: Telugu, Kannada, Malayalam
4. **Clinical Validation**: Accuracy assessment with palliative care experts

9.2 Medium-Term (Q3-Q4 2026)

1. **Multimodal RAG**: Process medical images (X-rays, medication photos)
2. **Personalization**: Condition-specific conversation flows
3. **Telemedicine Integration**: Connect to palliative care specialists
4. **Offline Mode**: Limited functionality for poor connectivity

9.3 Long-Term (2027+)

1. **Speech Biomarkers**: Detect distress from voice patterns
2. **Proactive Outreach**: Scheduled wellness check-ins
3. **Global Expansion**: Adaptation for 10+ countries
4. **Research Platform**: Anonymized data for palliative care research

9.4 Technical Improvements

Improvement	Current	Target
Response Latency	2-3s	<1s
Language Support	6	15+
Retrieval Accuracy	~85%	>95%
Entity Coverage	18 types	30+ types
Knowledge Graph Nodes	~500	10,000+

---

10. Current Limitations

10.1 Technical Limitations

1. **Latency**: Real-time voice has 2-3 second delay in fallback mode
2. **Offline Access**: Requires internet connectivity
3. **Audio Quality**: Performance degrades with poor audio input
4. **Complex Queries**: Multi-hop reasoning not fully implemented

10.2 Content Limitations

1. **Corpus Scope**: Limited to curated palliative care guidelines
2. **Regional Variation**: May not cover region-specific practices
3. **Drug Availability**: Medication recommendations may not reflect local availability
4. **Language Nuance**: Medical terminology translation may vary

10.3 Ethical Limitations

1. **Not a Replacement**: Cannot replace human healthcare providers
2. **Emergency Limitations**: Not equipped for emergency response
3. **Individual Variation**: Cannot account for individual patient factors
4. **Diagnostic Limitations**: Cannot diagnose conditions

10.4 Known Challenges in LMICs

1. **Digital Divide**: Access barriers for poorest populations
2. **Infrastructure**: Unreliable electricity and internet
3. **Trust**: Building trust in AI-based health information
4. **Regulation**: Varying regulatory requirements across countries

---

11. Conclusion

Palli Sahayak represents a significant advancement in democratizing palliative care knowledge through AI. By combining state-of-the-art RAG techniques, real-time voice AI, and knowledge graph technology with a focus on multilingual accessibility, the system addresses critical barriers to palliative care information access in India and beyond.

As a Digital Public Good, Palli Sahayak offers a replicable, adaptable model for healthcare AI in low-resource settings. The open-source architecture, combined with comprehensive documentation and modular design, enables other organizations and countries to build upon this foundation.

The path forward requires continued collaboration with medical professionals, community health workers, patients, and technology partners to ensure the system meets real-world needs while maintaining the highest standards of safety and accuracy.

Key Takeaways:

1. **Voice-First AI** can overcome literacy and technology barriers
2. **Hybrid RAG** with Knowledge Graphs improves medical accuracy
3. **Digital Public Goods** can scale healthcare AI globally
4. **Multilingual support** is essential for health equity
5. **Human oversight** remains critical for medical AI systems

---

12. References

RAG and AI Systems

1. A Comprehensive Survey of Retrieval-Augmented Generation - arXiv, 2024
2. GraphRAG: Unlocking LLM Discovery on Narrative Private Data - Microsoft Research, 2024
3. From Local to Global: A Graph RAG Approach - arXiv, 2025

Voice AI in Healthcare

4. How Generative AI Voice Agents Will Transform Medicine - PMC, 2025
5. AI Voice Agents in Healthcare - Twixor, 2025

Knowledge Graphs in Medicine

6. Leveraging Medical Knowledge Graphs Into LLMs - JMIR AI, 2025
7. PrimeKG: A Multimodal Knowledge Graph for Precision Medicine - Nature Scientific Data, 2023

Multilingual Healthcare AI

8. L2M3: Multilingual Medical LLM for Low-Resource Regions - arXiv, 2024
9. NVIDIA NIM for Hindi Language Healthcare - NVIDIA, 2025

Digital Health and Palliative Care

10. Ethical Challenges in AI Integration in Palliative Care - ScienceDirect, 2024
11. Digital Health Priorities for Palliative Care Research - PMC, 2022
12. Application of AI in Palliative Care: Bibliometric Analysis - Frontiers, 2025

Digital Public Goods

13. Digital Public Goods Alliance - DPGA
14. UN Sustainable Development Goals

Appendix A: API Reference

A.1 Core Endpoints

```
GET  /health                    - System health check

POST /api/query                 - RAG query

POST /api/bolna/query           - Bolna voice integration

POST /api/bolna/webhook         - Bolna event webhook

GET  /api/kg/health             - Knowledge graph health

POST /api/kg/query              - Knowledge graph query

GET  /api/kg/treatments/{symptom} - Find treatments

WS   /ws/voice                  - WebSocket voice streaming
```

A.2 Example Query

```
curl -X POST http://localhost:8000/api/query \

  -H "Content-Type: application/json" \

  -d '{

    "question": "What is the recommended treatment for cancer pain?",

    "language": "en",
```

```
  "include_sources": true
}'
```

Appendix B: Deployment Guide

B.1 Environment Variables

```
# Core
GROQ_API_KEY=your-groq-api-key

# Twilio (WhatsApp)
TWILIO_ACCOUNT_SID=your-sid
TWILIO_AUTH_TOKEN=your-token
TWILIO_WHATSAPP_FROM=whatsapp:+14155238886

# Bolna.ai
BOLNA_API_KEY=your-bolna-api-key
BOLNA_AGENT_ID=your-agent-id
BOLNA_WEBHOOK_SECRET=your-webhook-secret

# Google (Gemini Live)
GOOGLE_CLOUD_PROJECT=your-project-id
GEMINI_API_KEY=your-api-key

# Neo4j (Knowledge Graph)
NEO4J_URI=bolt://localhost:7687
NEO4J_USER=neo4j
NEO4J_PASSWORD=your-password
```

B.2 Docker Deployment

```
FROM python:3.11-slim
```

```
WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["uvicorn", "simple_rag_server:app", "--host", "0.0.0.0", "--port", "8000"]
```

Document Version: 1.0 **Last Updated**: December 2025 **Authors**: Palli Sahayak Development Team **License**: MIT License

---

Palli Sahayak: Compassionate AI for Palliative Care