AI-Driven Public Health Chatbot for Disease Awareness

Prepared for: Sanskar Gohil — Date: 2025-09-10

AI-Driven Public Health Chatbot for Disease Awareness
Prepared for: Sanskar Gohil

1. Project Summary

"AI-Driven Public Health Chatbot for Disease Awareness" is a mobile-first application that provides verified, actionable, and localized disease awareness and prevention information to the public. The system uses a Retrieval-Augmented Generation (RAG) pipeline combined with cloud LLM capability (OpenAI/Gemini) for high-quality responses. It emphasizes safety, misinformation blocking, gamified awareness scoring, analytics, and accessible design.

2. Why a Chatbot?

- Chatbots provide **interactive, conversational access** to health information, which is easier and more engaging than static websites or PDFs.
- They can deliver **personalized answers** instead of generic content.
- A chatbot allows **24/7 availability**, making health awareness information accessible anytime.
- It supports **multi-language extension**, ensuring wider reach in diverse populations.
- Unlike typical apps, chatbots mimic human conversation, which increases trust and adoption in awareness campaigns.

3. Why No Training from Scratch (NLP/ML)?

- Training an NLP/ML model from scratch requires **large datasets, GPUs, and months of training**, which is not feasible for student or prototype projects.
- Modern pre-trained LLMs (like GPT-4o, Gemini) already understand natural language deeply. We only need to **guide them with custom knowledge (RAG)**.
- **RAG (Retrieval-Augmented Generation)** lets us provide context (verified health guidelines) to the model, ensuring accuracy without retraining.
- This makes the project faster, more cost-effective, and practical while still using advanced AI techniques.

4. Goals & Use Cases

- Provide accessible, reliable public health awareness content (nutrition, hygiene, disease prevention, vaccination info, symptoms awareness).
- Prevent medical advice or prescriptions; surface education and refer to professionals for emergencies.
- Encourage healthy behavior via daily tips, quizzes, and awareness scoring.
- Collect anonymized analytics for public health insights (consent-based).

5. Core Features

- Domain-restricted conversational assistant (Health Awareness only)
- Filtering logic (intent classifier / keyword filter)
- Custom Knowledge Base (WHO, CDC, Govt health guidelines) stored in Vector DB (RAG)
- Cloud LLM integration (OpenAI/Gemini)
- Misinformation detection & correction
- Gamified Health Awareness Score & quizzes

- Category-based tips and daily tip notifications
- Analytics dashboard (admin) with anonymized trends
- Privacy-first design (anonymous mode, data minimization)

6. Unique & High-Value Differentiators

- Strict domain enforcement via multi-layer controls (prompt + classifier + post-filter)
- Localized content: region-specific prevalence, language support, and tips
- Misinformation blocker trained on curated myths dataset
- Awareness score, quizzes, and progress visualization (charts)
- Emergency flagging: clear referral to health services and helplines
- Exportable reports for educators/health workers (consent-based)

7. Technical Architecture

User Device (Mobile App - React Native or PWA)
  - Chat UI, Category Sidebar, Charts, Local storage for offline tips
  - Push notifications, Voice input (optional), BMI calculator

Backend (Node.js + Express / FastAPI)
  - Routes: /chat, /classify, /history, /admin/analytics
  - Filtering layer: intent classifier (zero-shot or small fine-tuned model)
  - RAG layer: Vector DB (Chroma / Pinecone / Weaviate) + Retriever
  - Model layer: Cloud LLM (OpenAI/Gemini API)
  - Business logic: scoring engine, quiz engine, misinformation detector
  - Database: MongoDB Atlas (user history, anonymized analytics, content store)

Vector DB (Chroma / Pinecone / Weaviate)
  - Stores embeddings for verified health docs (chunked)
  - Retriever returns top-k contexts for RAG

Admin Dashboard (React)
  - Upload content, monitor analytics, tune categories, view flagged queries

8. Tech Stack Recommendation

Frontend (Mobile-first):
  - React Native with Expo (recommended) OR React + PWA
  - UI: Tailwind CSS for web / NativeWind for React Native
  - Charts: Recharts or Victory Native
Backend & Services:
  - Node.js + Express or Python FastAPI
  - Embeddings: OpenAI embeddings or sentence-transformers
  - Vector DB: Chroma (self-host) or Pinecone (managed)
  - Cloud LLM: OpenAI (gpt-4o/gpt-4) or Google Gemini
  - Database: MongoDB Atlas
Deployment:
  - Frontend: Vercel (PWA) / Expo + Play Store (mobile)
  - Backend: Render / Railway / Heroku
  - Vector DB: Managed (Pinecone) or self-hosted on a small VM
Security:
  - HTTPS, API key vaulting (secrets manager), rate-limiting, input sanitization

9. Data Preparation & Knowledge Base

- Curate authoritative sources: WHO, CDC, Indian Ministry of Health, peer-reviewed summaries.
- Convert documents to text, chunk into 500–800 token chunks, create embeddings.
- Store chunks + metadata (source, date, language, region) in Vector DB.
- Maintain “myths and facts” dataset to detect and correct misinformation.

10. Filtering & Safety Pipeline
----
1) Receive user query.
2) Run quick keyword filter and zero-shot classifier to detect health-related intent or emergency content.
3) If emergency or harmful intent → immediately return safety response advising professional care.
4) If not health-related → reply with domain-restriction message.
5) If health-related → retrieve relevant KB context (top-k) and call model (cloud LLM) with system prompt + context.
6) Post-filter model response for hallucinations and policy compliance.
7) Log anonymized query for analytics (with consent).

11. System Prompts (Examples)

System prompt (RAG-enabled):
"You are a public health awareness assistant. Use only the provided context (if any) and general public health knowledge to answer. Do not provide medical diagnoses or prescribe medication. If the user describes an emergency or requests medical treatment, instruct them to seek professional care immediately."

12. Implementation Plan (12 weeks)

Week 1–2: Project setup, UI prototypes (React Native / PWA), content curation
Week 3–4: Backend skeleton, chat routes, simple keyword filter
Week 5–6: Build custom KB, chunking & embeddings, integrate Vector DB
Week 7–8: Integrate cloud LLM, RAG prompt engineering
Week 9: Implement scoring, quizzes, and charts
Week 10: Analytics dashboard and admin content upload
Week 11: Testing, safety audits, user testing
Week 12: Deployment (Play Store / Vercel) and final demo

13. File Structure

```
/project-root
  /mobile-app (React Native)
  /backend
    server.js (or app.py)
    /controllers
    /services (rag, classifier, model-adapter)
    /data (health_chapters.json)
    /models (UserChats.js)
  /admin-dashboard
  /docs (privacy-policy.md, data-sources.md)
```

14. Deployment Notes

- Use environment variables for API keys, use rate limiting and request quotas to avoid misuse.

- For Play Store release, follow data/privacy requirements and include clear disclaimers that the app is educational, not a medical service.

15. Testing & Evaluation

- Unit tests for backend routes, integration tests for RAG responses.
- Human evaluation of answer quality.
- Safety tests: ensure emergency phrases trigger referrals.
- Accessibility testing (screen reader support, large fonts).

16. Extras / Future Work

- Multi-language support (Hindi, local languages)
- Integrate local helpline lookup by region
- Partner with local health authorities for verified content updates
- Publish anonymized analytics for public health researchers

17. Appendix: Quick API Examples

/chat (POST)
Request JSON:
```
{ "userId":"anon123", "message":"Tell me diabetes prevention tips" }
```
Response JSON:
```
{ "reply":"...", "source":"WHO guideline" }
```

18. Contact & Credits

Prepared by: AI Project Planner (for Sanskar Gohil)
Date: 2025-09-10

--- End of document ---