**AI-driven public-health chatbot**

**What it is:** A conversational system (chatbot) that uses artificial intelligence to understand user questions about health and give tailored responses (disease possibilities, home-care advice, alerts).  
**Why it matters:** Core product — automates scalable health guidance and myth-correction.  
**Example:** A user types “I have fever and sore throat” → chatbot suggests likely causes, triage advice, and nearby emergency contacts.

**Natural Language Processing (NLP)**

**What it is:** A set of techniques and models that let computers understand, interpret, and generate human language.  
**Why it matters:** The chatbot’s ability to understand messages (intent) and extract facts (entities) depends on NLP.  
**Note:** In practice this includes tokenization, embeddings, intent classification, entity extraction, and response generation.

**Intent**

**What it is:** A label representing the user’s goal or purpose in a message (e.g., report\_symptom, ask\_vaccine\_info, request\_emergency).  
**Why it matters:** Intent detection drives which action the bot should take (answer, ask follow-ups, call emergency contacts).  
**Example:** “Where can I get vaccine?” → intent = ask\_vaccine\_location.

**Entity**

**What it is:** Key pieces of information extracted from user text (disease names, symptoms, location, dates).  
**Why it matters:** Entities let the bot fill slots for actionable responses (e.g., extracting “fever” and “3 days”).  
**Example:** From “I’ve had fever for 3 days” → entity symptom: fever, duration: 3 days.

**Rasa**

**What it is:** Open-source framework for building conversational assistants, with NLU + dialogue management.  
**Why it matters:** Good fit if you want full control, on-prem options, custom flows and multilingual pipelines for the chatbot.  
**Note:** Rasa separates NLU (intents/entities) from dialogue policies (how the bot responds).

**Dialogflow**

**What it is:** Google Cloud’s conversational AI platform that handles intent/entity detection and integrations.  
**Why it matters:** Easier to start with and integrates with Google ecosystem and telephony, but less flexible than some open-source stacks for heavy customization.

**Hugging Face**

**What it is:** Company & ecosystem providing transformer models, libraries (transformers) and a model hub (pretrained models).  
**Why it matters:** Source for multilingual transformer models (mBERT, XLM-R) and fine-tuning tools to build robust NLU for many languages.

**mBERT (Multilingual BERT)**

**What it is:** A multilingual version of BERT pretrained on Wikipedia in many languages; useful for multilingual understanding.  
**Why it matters:** Can help the chatbot understand multiple Indian languages without training one model per language.  
**Caveat:** Performance varies by language and may need fine-tuning.

**XLM-R (XLM-RoBERTa)**

**What it is:** A cross-lingual RoBERTa model pretrained on large multilingual corpora; often outperforms older multilingual models.  
**Why it matters:** Strong choice for building multilingual intent and entity detectors with better cross-language transfer.

**FastAPI**

**What it is:** A modern, high-performance Python web framework for building APIs (async capable).  
**Why it matters:** Useful for serving model inference endpoints and REST APIs that your frontend / WhatsApp webhook will call.

**Flask**

**What it is:** A minimal Python web framework for building web apps and APIs.  
**Why it matters:** Lightweight and easy to prototype; teams often start with Flask then move to FastAPI for high throughput.

**PostgreSQL**

**What it is:** Open-source relational database (ACID compliant) for structured data (user records, logs, reports).  
**Why it matters:** Good choice for storing historical user data, periodic reports, and structured business data.

**Redis**

**What it is:** In-memory data store used for caching, session storage, rate-limiting, and pub/sub patterns.  
**Why it matters:** Speeds up responses (cache frequent lookups), stores short-lived session context (dialog state), and supports real-time features.

**WhatsApp / SMS APIs**

**What it is:** Message-sending and receiving APIs (e.g., WhatsApp Business API, Twilio SMS) that let bots talk to users on messaging channels.  
**Why it matters:** Primary user channel for a public-health bot — WhatsApp for smartphone users, SMS for basic phones.

**USSD (Unstructured Supplementary Service Data)**

**What it is:** A session-based mobile network protocol (example: \*123#) used on feature phones for interactive menus.  
**Why it matters:** Enables service access without internet — important for rural users and feature phones.

**IVR (Interactive Voice Response)**

**What it is:** Automated phone systems that respond to caller input (DTMF or voice) and provide prerecorded or dynamic audio menus.  
**Why it matters:** Lets non-literate or low-tech users access information by phone voice prompts.

**Cloud hosting**

**What it is:** Running your servers and services on cloud providers (AWS, GCP, Azure) instead of physical machines you manage.  
**Why it matters:** Provides elasticity, managed services, and easier scaling for spikes (e.g., outbreaks).

**Kubernetes**

**What it is:** Container orchestration system that automates deployment, scaling, and management of containerized applications.  
**Why it matters:** Useful when you need reliable autoscaling, rolling updates, and resilient deployments for production chat services.

**Serverless (Functions as a Service)**

**What it is:** Cloud model where you run short-lived functions (e.g., AWS Lambda) and pay per execution rather than managing servers.  
**Why it matters:** Cost-efficient for spiky workloads or small microservices (notifications, webhook handlers).

**Anonymization & Encryption (privacy-first)**

**What it is:**

* **Anonymization:** Removing or modifying personal identifiers so data cannot be linked to an individual.
* **Encryption:** Cryptographically protecting data at rest and in transit so only authorized parties can read it.  
  **Why it matters:** Health data is sensitive — these practices reduce legal & ethical risks and support user trust.

**Real-time outbreak map (geospatial visualization)**

**What it is:** A live map that visualizes case reports or alerts based on user location or aggregated data.  
**Why it matters:** Visual and immediate awareness of local disease activity; useful for targeted alerts and resource planning.  
**Privacy note:** Must aggregate or anonymize location data to avoid exposing individuals.

**Continuous learning (online / incremental learning)**

**What it is:** Processes that let models be retrained or updated regularly using new labeled data (feedback, newly verified cases).  
**Why it matters:** Keeps NLU accurate as language, slang, or disease patterns change; helps adapt to new symptoms/terms.  
**Caveat:** Must monitor for data drift and ensure safe, validated updates.

**Multilingual support (local languages)**

**What it is:** Ability for the bot to understand and reply in multiple languages and dialects.  
**Why it matters:** Essential for reach and equity in India — increases accessibility and trust.

**Voice input / Speech-to-Text (ASR)**

**What it is:** Automatic Speech Recognition converts spoken words into text the bot can process.  
**Why it matters:** Enables hands-free and low-literacy access when combined with TTS (text-to-speech) for replies.

**Government health databases (integration)**

**What it is:** Authoritative data sources (APIs or published data) from public health agencies used for case counts, advisories, and verified information.  
**Why it matters:** Ensures the bot’s factual answers and outbreak maps are up-to-date and trustworthy.

**Monitoring & Analytics**

**What it is:** Tools and dashboards that track bot usage, intent accuracy, user satisfaction, errors, and health-outcome metrics.  
**Why it matters:** Needed to measure impact (e.g., awareness rise), detect failures, and guide improvements.

**Pilot deployment**

**What it is:** A small-scale field test of the system with limited users/area before full roll-out.  
**Why it matters:** Validates assumptions, discovers real-world issues (connectivity, language edge cases), and tunes the system.

**Accuracy (model metrics like accuracy, precision, recall, F1)**

**What it is:** Quantitative measures of model performance. Accuracy = overall correct predictions; precision/recall/F1 measure class-specific behavior.  
**Why it matters:** The slide mentions ≥80% accuracy — good as a target but choose metrics (F1, recall) appropriate to critical intents (e.g., emergency detection should minimize false negatives).

**Quick next steps:**

* Turn this into a printable glossary (CSV, PDF or slide) for your SIH submission.
* Expand any single term into implementation steps (e.g., “How to set up Rasa + FastAPI + Redis for this chatbot”).  
  Tell me which format you want and I’ll produce it right away — or I can generate the glossary file now if you prefer.

**Proper Usage in Your Chatbot Project**

**1. XLM-R (XLM-RoBERTa)**

* **Usage:**
  + Train / fine-tune for **intent classification** (e.g., “report symptom”, “ask about vaccine”).
  + Extract **entities** (e.g., symptoms, disease names, locations).
  + Works well across **multiple Indian languages** without training a separate model per language.
* **How:**
  + Use Hugging Face transformers library.
  + Fine-tune XLM-R on a labeled dataset of multilingual queries.
  + Deploy as the NLU model behind your chatbot.

**2. mBERT (Multilingual BERT)**

* **Usage:**
  + Similar role as XLM-R, but lighter.
  + Good for **resource-constrained environments** where you can’t afford the compute of XLM-R.
  + Acts as a **baseline multilingual NLU model** before upgrading to XLM-R.
* **How:**
  + Fine-tune for intent/entity tasks.
  + Can be used in Rasa pipelines via Hugging Face integration (LanguageModelFeaturizer).

**3. RASA NLU**

* **Usage:**
  + The **engine** that ties everything together for natural language understanding.
  + Handles **intent classification, entity extraction, dialogue management**.
  + Allows building conversation flows, fallback handling, slot filling.
* **How:**
  + Plug in XLM-R or mBERT as a **featurizer/encoder** inside the Rasa NLU pipeline.
  + Configure **domain.yml** for intents/entities.
  + Train on your custom dataset (health queries).

**4. NLP (Natural Language Processing)**

* **Usage:**
  + Umbrella term — every text operation in the chatbot is NLP.
  + Covers **preprocessing** (tokenization, normalization, transliteration for Indian languages),  
    **understanding** (intent/entity via models),  
    **generation** (responses, possibly multilingual TTS).
* **How:**
  + Before feeding text to XLM-R/mBERT, clean & normalize input (remove spelling noise, handle Hinglish).
  + Use NLP techniques like lemmatization, synonym mapping, translation if needed.

**5. Redis**

* **Usage:**
  + **Session storage:** Store user conversation state between turns (short-term memory).
  + **Caching:** Cache frequent queries (e.g., “What are COVID symptoms?”) to answer instantly.
  + **Queue / PubSub:** Manage real-time updates (alerts, notifications).
* **How:**
  + Connect Redis with Rasa’s Tracker Store to persist conversation history.
  + Use as a fast cache for outbreak data or health alerts.

**6. Kubernetes**

* **Usage:**
  + **Orchestrates your microservices** (chatbot NLU service, FastAPI backend, database, Redis, frontend).
  + Handles **scaling** (more pods when traffic spikes, e.g., outbreak),  
    **rolling updates** (deploy new model without downtime),  
    **auto-healing** (restarts crashed containers).
* **How:**
  + Package chatbot services in **Docker containers**.
  + Deploy to Kubernetes (cloud or on-prem).
  + Use Kubernetes features like **Horizontal Pod Autoscaler** to scale NLP models dynamically.

**🚀 Putting It All Together (Usage Flow)**

1. **User query** (WhatsApp, SMS, IVR) → text.
2. **NLP preprocessing** → clean input (tokenization, normalization).
3. **Rasa NLU pipeline** calls **XLM-R / mBERT** → extract intent + entities.
4. **Rasa Core** → decides next step (ask question, show advice, send alert).
5. **Redis** → stores conversation state + caches frequent responses.
6. **FastAPI/Flask backend** → fetches outbreak data / generates reports.
7. **Kubernetes** → ensures all these services stay up, scale automatically during peak loads.