🎯 Prototype Goal

A **multilingual Q&A chatbot** that:

- Takes queries in **English/Hindi/1 local language (Gujarati or Marathi)**.
- Uses **Rasa** for intent detection & context flow.
- Uses **XLM-R** (or IndicBERT fallback) for multilingual embeddings/understanding.
- Uses **LangChain + Vector DB (FAISS)** for RAG (retrieving answers from uploaded PDFs/FAQs).
- Responds back in the user’s language.

✅ This matches the **Problem Statement** and is feasible in hackathon time.

🛠️ Prototype Architecture (Backend Focus)

1. Input (Student Query)

- Text (voice can be future scope).
- Supported: English, Hindi, Gujarati (demo 3 languages).

2. Language Understanding (XLM-R)

- Converts query → embeddings (language-agnostic).
- Handles multilingual intent detection (via Rasa + XLM-R).

3. Conversation Manager (Rasa)

- Detects **intent** (FAQ, timetable request, deadline, fallback).
- Maintains **context** across multiple turns.

- Routes query:
    - If FAQ → returns from stored FAQ DB.
    - Else → triggers RAG.

4. Retrieval-Augmented Generation (LangChain)

- **LangChain document loader** → parse uploaded PDFs/circulars.
- Store embeddings in **FAISS Vector DB**.
- Query embedding matched against DB.
- Return top document chunks.

5. Response Generation

- Simple template-based response (not heavy LLM in hackathon).
- Translate back into student's input language (using IndicTrans / HuggingFace NLLB for multilingual).

6. Output

- Sent to frontend (Web UI / WhatsApp integration).

⚡ Backend Tech Stack

- **Rasa** → Intent + Dialogue Manager.
- **XLM-R (HuggingFace)** → Multilingual embeddings.
- **LangChain** → Orchestrates RAG pipeline.
- **FAISS (local vector DB)** → Store/retrieve doc embeddings.
- **Flask/FastAPI wrapper** → Expose API → frontend integration.

- **Translation model** → HuggingFace IndicTrans2 (lightweight).

📌 Implementation Steps for Hackathon

Phase 1 – Setup Core Pipeline

1. **Rasa Project Setup**
   - Define 5–6 intents: greeting, FAQ query, timetable query, deadline, fallback.
   - Train basic NLU with sample phrases (in English + Hindi + Gujarati).
2. **XLM-R Integration**
   - Use HuggingFace pipeline for embeddings.
   - Replace Rasa's default featurizer with multilingual embeddings (Rasa allows custom NLU components).
3. **LangChain + FAISS Setup**
   - Load 3–4 sample PDFs (mock circulars).
   - Parse into chunks (200–300 tokens).
   - Store embeddings in FAISS.
   - Build retrieval chain.
4. **Connect Rasa → LangChain**
   - If Rasa intent = "document_query" → trigger LangChain retrieval.
   - Return top answer chunk to Rasa.
5. **Translation Layer**
   - Detect input language (langdetect).

    - If not English → translate response back into input language.

6. **Expose API**
    - Wrap in FastAPI endpoint → `/chat` (input query → JSON response).
    - This is what frontend connects to.

Phase 2 – Testing & Demo Prep

- Upload **sample dataset**: 3 FAQs + 2 circulars (PDF timetables/deadlines).
- Test queries in English/Hindi/Gujarati.
- Demo scenario:
    - Student asks in Hindi → “मेरी परीक्षा का टाइमटेबल दिखाओ।”
    - Bot detects intent, fetches from uploaded timetable PDF.
    - Responds in Hindi → “यह रहा आपका टाइमटेबल: [PDF link].”

🚀 Deliverables for SIH PPT (Prototype)

1. **Block Diagram** (pipeline: Input → XLM-R → Rasa → LangChain RAG → Response).
2. **Working Demo** (on 3 languages, 3–4 docs).
3. **Screenshots** (console logs + frontend chat).
4. **Future Scope Slide** (voice, SMS, campus nav, dashboards, etc.).

🎯 What You Need to Implement (Backend Role)

- Rasa NLU + dialogue setup.
- Custom NLU component → plug XLM-R embeddings.
- LangChain + FAISS doc retrieval pipeline.
- Integration between Rasa → LangChain → Response.
- Translation module for multilingual replies.
- Expose APIs via FastAPI.

👉 This way, you will have a **solid, working prototype** that:

- Proves **multilingual + PDF retrieval**.
- Stays within **SIH problem statement**.
- Leaves space for **future features** later.