**News API-Based RAG Alert System with WhatsApp Integration**

**Project Plan(Tentative)**

**1. Project Overview**

* Fetches news articles from APIs (NewsAPI, Social Media /external apis etc.)
* Processes and categorizes news using a Retrieval-Augmented Generation (RAG) pipeline.
* Classifies news into priority levels (Urgent, High, Moderate, Low).
* Sends alerts via WhatsApp.

**2. Development Plan**

**Phase 1: Setup & Data Collection**

**1.1 Database Setup**

* Install PostgreSQL/MySQL or Any Cloud Based RDB
* Design database schema to store news articles and classification results.
* Implement SQLAlchemy ORM for database interactions.

**1.2 Fetch News Data**

* Integrate NewsAPI / Social Sites using External apis.
* Implement a Python script that periodically retrieves and updates news data using scheduled API requests.
* Store raw news data in the database.

**1.3 Automate News Fetching**

* Implement cron jobs for scheduling tasks.
* Log API responses for debugging.

**Phase 2: RAG Pipeline & NLP Processing**

**2.1 Text Preprocessing**

* Clean and preprocess text (remove stop words, lemmatization, etc.).
* Convert text into a standardized format.

**2.2 Embeddings & Vector Database**

* Convert news articles into embeddings using Sentence Transformers.
* Store embeddings in FAISS for efficient retrieval.

**2.3 RAG Query Implementation**

* Integrate FAISS with Llama 3.2/DeepSeek 8b for retrieval-based AI responses.(Using Groq Based API)
* Implement a query engine to fetch relevant news.
  1. **Summarization & Sentiment Analysis**
* Use Llama 3.2/ DeepSeek 8b for summarizing news articles. .(Using Groq Based API)
* Perform sentiment analysis (positive, negative, neutral).
* Store results in the database.

**Phase 3: Alert Classification & Priority Flags**

**3.1 Flag Classification System**

* Define classification rules based on keywords, sentiment, and severity.
* Train an ML model to categorize news.
* Store priority flags in the database.

**3.2 Implement Alert Decision Logic**

* Define rules for sending WhatsApp alerts.
* Ensure alerts are triggered only for urgent and high-priority news.
* Log alerts to prevent duplicate notifications.

**Phase 4: WhatsApp Integration & Alert Triggering**

**4.1 Setup WhatsApp API**

* Configure Twilio API / Meta Cloud API.(Need to check for the charges)
* Obtain API credentials for WhatsApp messaging.

**4.2 Implement WhatsApp Alert System**

* Format WhatsApp messages for readability.
* Send alerts based on flag classification.
* Log sent alerts for monitoring.

**4.3 Test & Optimize Alert System**

* Test message formatting and delivery.
* Ensure rate limits do not block messages.
* Debug and optimize the alerting mechanism.

**Phase 5: Deployment & Automation**

**5.1 Deploy Backend**

* Set up a FastAPI-based backend
* Deploy on AWS/GCP/Azure using Docker.
* Deploy on Cloud Server

**3. Summary & Timeline (Tentative)**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Duration** | **Key Deliverables** |
| Setup & Data Collection | Week 2 | Database, News API Integration, Automated Fetching |
| RAG Pipeline & NLP | Week 2-4 | Llama 3.2, Summarization & Sentiment Analysis |
| Alert Classification | Week 4-5 | News Categorization, Priority Flags, Decision Logic |
| WhatsApp Integration | Week 5-6 | Twilio API, WhatsApp Alerts, Testing |
| Deployment & Testing | Week 6-8 | Cloud Deployment on Server |

**Tools & Technologies Used:**

* **Backend:** Python, FastAPI, SQLAlchemy
* **Database:** PostgreSQL/MySQL, Or Any Cloud Based RDB FAISS Optional
* **NLP & AI:** Llama 3.2,DeepSeek, Sentence Transformers, Hugging Face,
* **Messaging:** Twilio API, Meta WhatsApp API
* **Deployment:** Docker,AWS/GCP/Azure

**4. FLOW CHART (Tentative)**

