1. Introduction

This document outlines the process of training and deploying an open-source LLM-powered event recommendation system that assists users in selecting events based on their preferences and notifies them via WhatsApp/Telegram when relevant events occur.

2. System Overview

2.1 Functional Requirements

Users interact via WhatsApp/Telegram.

The system collects user preferences (event type, location, budget, etc.).

It scrapes or fetches event data from online sources (Eventbrite, Meetup, etc.).

Uses an LLM to match and recommend events.

Sends real-time notifications when relevant events are found.

2.2 Architecture Overview

Data Collection: Scrapes event data from APIs or websites.

User Preference Storage: Stores preferences in a database.

Event Matching Engine: Uses a vector database for search.

Notification System: Sends alerts via WhatsApp/Telegram.

Chatbot Interface: Uses an LLM to handle user queries.

3. Training the LLM

3.1 Selecting a Base Model

An open-source model like Mistral-7B, LLaMA-2 (7B), or Falcon-7B will be used as the foundation.

3.2 Fine-Tuning Data Preparation

The model is trained using:

Historical Event Data: Event descriptions, tags, locations.

User Preference Data: Example conversations mapping user inputs to event selections.

Contextual Conversations: Sample WhatsApp/Telegram chat interactions.

3.3 Fine-Tuning Process

Using LoRA (Low-Rank Adaptation) or QLoRA to train the LLM efficiently:

Prepare Dataset: Convert conversations into structured JSON.

Tokenization: Process text data using SentencePiece or Tokenizers.

Fine-tune the model: Train using Hugging Face’s Transformers + PEFT.

Evaluation: Test on unseen queries.

3.4 Using RAG (Retrieval-Augmented Generation)

To ensure real-time updates, the model will retrieve live event data from a vector database (FAISS/ChromaDB) instead of relying on static training data.

4. Deployment Pipeline

4.1 Backend Setup

FastAPI for handling event search queries.

Celery + Redis for background task scheduling.

PostgreSQL for storing user preferences.

FAISS/ChromaDB for vector-based event search.

4.2 Event Data Collection

APIs: Eventbrite, Ticketmaster, Meetup.

Web Scraping: Scrapy for extracting event listings.

Scheduler: Celery task runs every 6 hours to refresh event data.

4.3 Chatbot & Notifications

WhatsApp Integration: Twilio API for sending messages.

Telegram Bot: Uses Telegram Bot API.

Webhook Listener: Receives user messages and routes them to the LLM.

4.4 Hosting & Scaling

Model Hosting: Hugging Face Inference API or ONNX Runtime.

Cloud Infrastructure: Deploy on AWS Lambda/Fly.io/Railway.

Monitoring: Prometheus + Grafana for logs and metrics.

5. Future Enhancements

Voice-based event discovery using Whisper/GPT-SoVITS.

Personalized ranking models to improve recommendation quality.

Integration with Google Calendar to sync user schedules.

Conclusion

This pipeline enables real-time event recommendations using an LLM, vector search, and automated notifications via WhatsApp/Telegram. The combination of RAG-based retrieval and user feedback loops ensures accuracy and relevance in recommendations.