Project Overview

This project aims to develop an **AI-powered console-based chatbot** using C++ that can process real-time data, learn from past conversations, fetch information from the internet, and interact with users via text. The chatbot will continuously improve responses based on user interactions and integrate external APIs for enhanced functionality.

Objectives

- Develop a self-learning chatbot that adapts to user inputs.
- Implement real-time data fetching from APIs (e.g., news, weather, Wikipedia).
- Enhance chatbot responses using Natural Language Processing (NLP).
- Enable voice-based interaction with speech-to-text (STT) and text-to-speech (TTS) features.
- Store and analyze past conversations for personalized responses.

Key Features

Core Functionalities

- **Adaptive Conversations:** The chatbot can remember past interactions and improve its responses over time.
- **Personality Modes:** Users can choose from different chatbot personalities (e.g., Friendly, Formal, Sarcastic).
- **Real-Time Data Fetching:** The chatbot can retrieve live data (e.g., weather, news, definitions) via API calls.
- **Self-Learning Mechanism:** Uses machine learning techniques like Markov Chains to refine responses.
- **Offline Mode:** Limited functionalities when there is no internet connection.

Speech & AI Integration

- **Speech-to-Text (STT):** Users can talk to the chatbot instead of typing.
- **Text-to-Speech (TTS):** The chatbot can read responses aloud.

- **AI-Based Learning:** Incorporates basic sentiment analysis and text prediction to improve user engagement.

User Interaction

- Users can ask general questions and receive intelligent responses.
- Users can request real-time data like weather updates or news headlines.
- The chatbot will store and analyze previous conversations for better engagement.
- Users can teach the chatbot new phrases to expand its knowledge base.

Technologies & Tools

Programming Language & Libraries

- **C++** (Core Development)
- **regex** (Natural Language Processing)
- **fstream** (File Handling for Memory System)
- **unordered_map & vector** (Data Storage & Retrieval)
- **Markov Chains** (Self-Learning AI Model)
- **TF-IDF Algorithm** (Text Analysis)

APIs & Internet Data Fetching

- **cURL** (Fetching data from external APIs)
- **nlohmann/json.hpp** (Processing JSON responses)
- **OpenWeather API** (Weather data)
- **NewsAPI.org** (Latest news updates)
- **Wikipedia API** (Information retrieval)
- **OpenAI API** (For AI-based conversations)

Voice Processing

- **Google Speech API / CMU Sphinx** (Speech-to-Text Processing)
- **eSpeak / Windows SAPI** (Text-to-Speech Conversion)

Team Members & Responsibilities

Member	Role	Responsibilities
Mehin	Team Leader & Architect	Design architecture, oversee development, API integration
Anik	NLP & Data Scientist	Implement NLP processing, Markov Chains, TF-IDF model
Zehad	Backend Developer	File handling, chatbot learning system, database integration
Jubaer	API & Web Integration	Implement cURL, JSON handling, OpenWeather or NewsAPI integration
Nahid	Voice & AI Integration	Develop STT, TTS features, OpenAI API integration

---

Development Timeline

Month 01:

Research & Planning

- Define chatbot requirements and set up the development environment.
- Research APIs & NLP techniques.

Core Chatbot Development

- Implement basic chatbot responses.
- Develop chatbot memory system.

Month 02:

API & Data Fetching

- Integrate weather, news, and Wikipedia API.
- Develop real-time web fetching system.

AI & Learning Mechanisms

- Implement Markov Chains for self-learning.
- Apply TF-IDF for text analysis.

Month 03:

Speech Processing

- Develop speech-to-text (STT) system.
- Implement text-to-speech (TTS).

Testing & Debugging

- Unit testing chatbot functionalities.
- Fix errors and optimize performance.
- Finalize documentation & prepare for project submission.

Expected Challenges & Solutions

Challenge	Solution
Real-time data fetching issues	Use cURL with error handling
Slow learning process	Optimize Markov Chain implementation
Speech recognition accuracy	Train with a dataset for better recognition
API limitations	Use free-tier APIs and caching mechanisms

Expected Outcomes

- A **console-based chatbot** capable of **real-time data processing, self-learning, and AI-powered interactions**.
- An **adaptive and interactive** chatbot that improves responses based on user input.
- A **voice-enabled** chatbot for enhanced user experience.
- **Dynamic data fetching** from news, weather, and Wikipedia APIs.

Future Enhancements

- Convert the **console-based** into a **GUI application** (with Qt framework).
- Implement **database-based memory storage** instead of flat files.
- Develop a **mobile app version** with the same AI functionalities.

Conclusion

This AI-powered chatbot will showcase **real-world data processing, self-learning AI, and advanced API integrations** using C++. The project will enhance team skills in **NLP, AI, API handling, and speech processing** while delivering a useful and intelligent chatbot assistant. The successful completion of this project will demonstrate our proficiency in AI-based development and real-time data processing.

Team Member's

- **Md Mehin Islam (Team Leader)**
  Id: 20244203094
  CSE, BUBT
- **Anik Miah**
  Id: 20244203104
  CSE, BUBT
- **Md Zehad Ali**
  Id: 20244203141
  CSE, BUBT
- **Ehsanul Mahbub Jubaer**
  Id: 20244203142
  CSE, BUBT
- **Md Nahid Hasan**
  Id: 20244203109
  CSE, BUBT

Instructor Approval:

Signature: _____

Date: _____