# PromptBotBlazor Documentation

## Project Overview

**PromptBotBlazor** is a web application built using **Blazor Server** on the **.NET 8** framework. The goal of the project is to create a chatbot capable of responding to user queries based on predefined data, with the ability to generate new responses using Generative AI techniques. The project initially started as a Rule-Based Chatbot relying on a `data.csv` file to store questions and answers. It was later transformed into a Generative AI application by fine-tuning a language model (AraGPT2) and integrating it with the app.

---

## Project Objective

- **Initial Version**: Provide predefined responses based on data in `data.csv`. If a question is not found, prompt the user to manually input a response.
- **Final Version (Generative AI)**: Enable the generation of new responses for questions not present in `data.csv` using a fine-tuned language model (AraGPT2), while retaining the ability to store new responses for future use.

---

## Project Components

### 1. Blazor Application (`PromptBotBlazor`)

- **Main File**: `PromptBot.razor`
  - A user interface containing an input field for the question (Prompt) and a "Send" button to submit the question.
  - Displays responses in a "Result" section.
  - Includes logic to read data from `data.csv` and send requests to a local API for generating responses.
- **Data File**: `data.csv`
  - Stores questions and answers in CSV format (e.g., `"The product is great and easy to use","I'm glad you liked the product! Would you like to know more about its features?"`).
- **Used Libraries**:
  - `Microsoft.AspNetCore.Components.Web`: For running the Blazor application.
  - `System.Net.Http.Json`: For sending HTTP requests to the local API.

### 2. Generative AI Model (AraGPT2)

- **Model**: AraGPT2-base (from Hugging Face, library `aubmindlab/aragpt2-base`).

- **Training**:
  - Fine-tuning was performed on the project's data from `data.csv` after converting it into a text file (`train.txt`).
  - Training was done using the `transformers` library from Hugging Face.
- **Training File**: `fine_tune_aragpt2.py`
  - Loads the model, prepares the data, trains the model, and saves the fine-tuned model in the `aragpt2-finetuned` directory.

**3. Local API (`api.py`)**

- **Purpose**: Provides an interface for communication between the Blazor application and the fine-tuned model.
- **Framework**: Flask (a lightweight Python framework).
- **Functionality**:
  - Receives the question from the Blazor app via an HTTP request.
  - Uses the fine-tuned model (AraGPT2) to generate a response.
  - Returns the response to the app.
- **Endpoint**: Runs on `http://localhost:5000/generate`.

---

# Project Workflow

1. **Run the Local API**:
   - `api.py` is executed to load the fine-tuned model and handle requests.
2. **Run the Blazor Application**:
   - The app is launched using the `dotnet run` command.
3. **Enter a Question**:
   - The user types a question in the app's interface and clicks "Send".
4. **Search in Data**:
   - The app searches in the `promptResponses` dictionary loaded from `data.csv`.
   - If the question exists, the corresponding answer is displayed directly.
5. **Generate a New Response**:
   - If the question is not found, the app sends the question to the local API.
   - The API uses the fine-tuned AraGPT2 model to generate a new response.
   - The new response is stored in `data.csv` and displayed to the user.
6. **Manual Input (Optional)**:
   - If the API fails to generate a suitable response, the app prompts the user to manually input an answer.

---

# Setup and Installation

**Prerequisites**

- **Blazor Application**:
  - .NET 8 SDK.
  - Visual Studio or any editor (e.g., VS Code).
- **Generative AI Model**:
  - Python 3.9 or later.
  - GPU (optional but recommended for faster training).
  - Required libraries:
  - `pip install torch transformers datasets accelerate flask`

**Steps to Run the Project**

1. **Set Up the Model**:
   - Prepare your data in a `train.txt` file (as described in the training section).
   - Run the training script:
   - `python fine_tune_aragpt2.py`
   - The fine-tuned model will be saved in the `aragpt2-finetuned` directory.
2. **Run the Local API**:
   - Execute the `api.py` file:
   - `python api.py`
   - Ensure the API is running on `http://localhost:5000/generate`.
3. **Run the Blazor Application**:
   - Open the project directory (`PromptBotBlazor`).
   - Run the command:
   - `dotnet run`
   - Access the interface at `http://localhost:5187/promptbot`.
4. **Test the Application**:
   - Enter a question already in `data.csv` (e.g., "The product is great and easy to use").
   - Enter a new question (e.g., "How do I make a cake?") and observe the generated response.

---

# Challenges and Limitations

- **Data Size**:
  - If the training data is too small (e.g., fewer than 100 questions and answers), the model may suffer from overfitting (memorizing the data instead of learning).
- **Computational Resources**:
  - Training and running the model require a powerful GPU. Without a GPU, training will be very slow.
- **Response Quality**:

    - AraGPT2-base may not handle complex contexts or long questions well. For better responses, a larger model like AraGPT2-medium can be used (but it requires more resources).
- **Arabic Language**:
    - The model supports Arabic, but it may need further improvement for handling colloquial dialects (e.g., Egyptian dialect).

---

# Future Improvements

- **Enhance Response Quality**:
    - Add more diverse and larger training data.
    - Experiment with a larger model like AraGPT2-medium or LLaMA.
- **Support Partial Search**:
    - Modify the code to support partial search (e.g., if the user types part of a question, the app finds the closest match).
- **Improve Arabic Language Support**:
    - Enhance support for colloquial dialects by training the model on dialectal data (e.g., Egyptian Arabic).
- **Integrate with Other Interfaces**:
    - Add voice input support to improve the user experience.

---

# Project Timeline

- **Start**: The project began as a Rule-Based Chatbot relying solely on `data.csv`.
- **Initial Development**: Added the feature to request manual answers for missing questions.
- **Transition to Generative AI**:
    - First Attempt: Used Hugging Face Inference API (limited).
    - Final Solution: Fine-tuned AraGPT2 and ran it locally with an API.
- **Current Date**: April 22, 2025. The project now operates as a Generative AI with a fine-tuned model.

---