**Project Documentation: Multi-Persona ChatBot – LPU**

1. Abstract

This project presents the development of a Multi-Persona AI Chatbot capable of responding differently based on the selected role such as Doctor, Teacher, Tech Support, or Travel Agent. The chatbot integrates Cohere’s language model with a Flask backend and a modern, responsive HTML/CSS/JavaScript frontend. Persona-specific prompts ensure that the responses are context-aware and aligned with the selected role. The project demonstrates the use of Prompt Engineering, Natural Language Processing (NLP), and full-stack development to build an engaging and interactive AI system.

# 2. Introduction

Traditional chatbots often follow a single personality, limiting their adaptability to varied user needs. Multi-persona chatbots expand their scope by customizing interactions according to context and role, creating a more human-like, task-specific experience.

## Objective

To design and develop an AI chatbot that can switch between multiple personas and provide responses aligned with each role’s characteristics.

# 3. Problem Statement

Conventional AI chatbots fail to address specialized contexts effectively because they are not tuned for specific roles. This can lead to generic or irrelevant responses, reducing usability in professional or domain-specific applications.

# 4. Scope

- Support multiple roles: Doctor, Teacher, Tech Support, and Travel Agent.  
- Provide context-aware, persona-aligned replies.  
- Deploy locally with potential for cloud deployment.  
- Extendable to include more roles and integrate advanced AI models.

# 5. Literature Review

Existing systems like ChatGPT, Bard, and domain-specific bots show strong general performance but lack flexible role-switching in real-time. Role-specific fine-tuning or prompt engineering is still an emerging field, and Cohere’s API provides a robust platform for experimenting with such features.

# 6. Proposed System

## Features

- Persona selection dropdown.  
- Real-time response generation.  
- Dynamic prompt injection for persona alignment.  
- Friendly and modern UI with live feedback.

## Advantages

- Personalized user interaction.  
- Increased contextual accuracy.  
- Scalable for adding new personas.

# 7. System Design

## Workflow Diagram

[User] → [Frontend: HTML/CSS/JS] → [Flask Backend] → [Cohere API] → [Response] → [Frontend Display]

## ER Diagram

Entities:  
1. User (user\_id, name, message, selected\_persona)  
2. ChatSession (session\_id, user\_id, timestamp)  
3. Persona (persona\_id, persona\_name, prompt\_template)  
  
Relationships:  
- User can have multiple ChatSessions.  
- ChatSession is linked to one Persona.

# 8. Implementation

## Tech Stack

Backend: Flask (Python)  
Frontend: HTML, CSS, JavaScript  
AI/NLP: Cohere API  
Others: Prompt Engineering

## Code Highlights

- Persona dictionary storing role-specific preambles.  
- API endpoint '/chat' for processing user queries.  
- Dynamic JSON request handling in JavaScript.

# 9. Results

Successfully deployed a working multi-persona chatbot. Verified role-based contextual accuracy. Positive UI/UX with responsive design.  
  
Example Output:  
Persona: Doctor  
User: 'I have a headache for 3 days.'  
Bot: 'I’m sorry to hear that. Please rest, stay hydrated, and consult a physician if the pain persists.'

# 10. Testing

- Functional Testing: Checked API calls, persona switching, and error handling.  
- UI Testing: Verified responsive design across devices.  
- API Testing: Used sample queries to validate Cohere model output.

# 11. Conclusion

The project successfully demonstrates that prompt engineering with AI models can produce specialized, context-aware chatbot responses. The modular design allows easy expansion to include additional roles and advanced NLP features.

# 12. Future Enhancements

- Deploy to cloud platforms (Heroku, Render).  
- Add speech-to-text and text-to-speech features.  
- Integrate with a database for chat history.

# 13. References

Cohere API Documentation  
Flask Documentation  
MDN Web Docs – HTML/CSS/JavaScript  
Python Official Documentation