CodeCatalyst - 2025

TITLE PAGE

- **Problem Statement ID –PSM3**
- **Problem Statement Title-Create an adaptive digital self**
- **PS Category-** ***Machine Learning***
- **Team ID -CH32**
- **TeamName -JD**

JD

Create an adaptive digital self

TechSparks
CSE – Association
Department of Computer Science and Engineering

Proposed Solution – Slide Content

💡 Idea / Solution / Prototype

Adaptive Digital Self – AI Clone

An AI-poweredsystemthatlearnsanindividual's communication and writing style from textual data to generate personalized, human-like responses, ensuring data privacy and ethical AI usage.

Detailed Explanation of the Proposed Solution

The system uses Conversation.csv as the input dataset containing user dialogues or story text.
Applies Natural Language Processing (NLP) techniques to extract the writing pattern, tone, and vocabulary.
Uses TF–IDF Vectorization to convert text into numerical features and represent each line's unique linguistic structure.
Implements Cosine Similarity to identify the most contextually relevant sentence in the dataset when the user types a message.
Generates a response that closely matches the user's style — maintaining the same tone, grammar, and intent.
Operates fully offline, ensuring that no personal data is stored or shared externally.
The output is displayed as a real-time chatbot conversation, mimicking the user's natural language.

How It Addresses the Problem

Reduces the gap between generic chatbots and human-like communication.
Provides an ethical personalization framework — AI adapts to the user without collecting sensitive data.
Enables users to see how AI can reflect their own communication behavior safely and transparently.
Supports creative applications like story continuation, digital companions, or writing assistants.

Innovation and Uniqueness

Learns directly from the user's own dataset (Conversation.csv) instead of using internet data.
Uses lightweight, explainable ML models instead of heavy neural networks — fast and transparent.
Demonstrates privacy-preserving AI cloning, ensuring full local data control.
Can adapt to any genre or tone — formal, conversational, thriller, or emotional — depending on dataset style.
Easy to implement, runs in IDLE, and suitable for real-time demo within 5 hours.

JD

TECHNICAL APPROACH

TechSparks
CSE – Association
Department of Computer Science and Engineering

Technologies to be Used

Component Tool / Technology
Programming Language Python
IDE IDLE / Jupyter Notebook Data Handling Pandas Machine Learning / NLP Scikit- learn (TF–IDF Vectorizer, Cosine Similarity)

Text Preprocessing String Cleaning, Stopword Removal
Visualization / Interface Console-based Chat Interface
Dataset Conversation.csv (user's own text data)

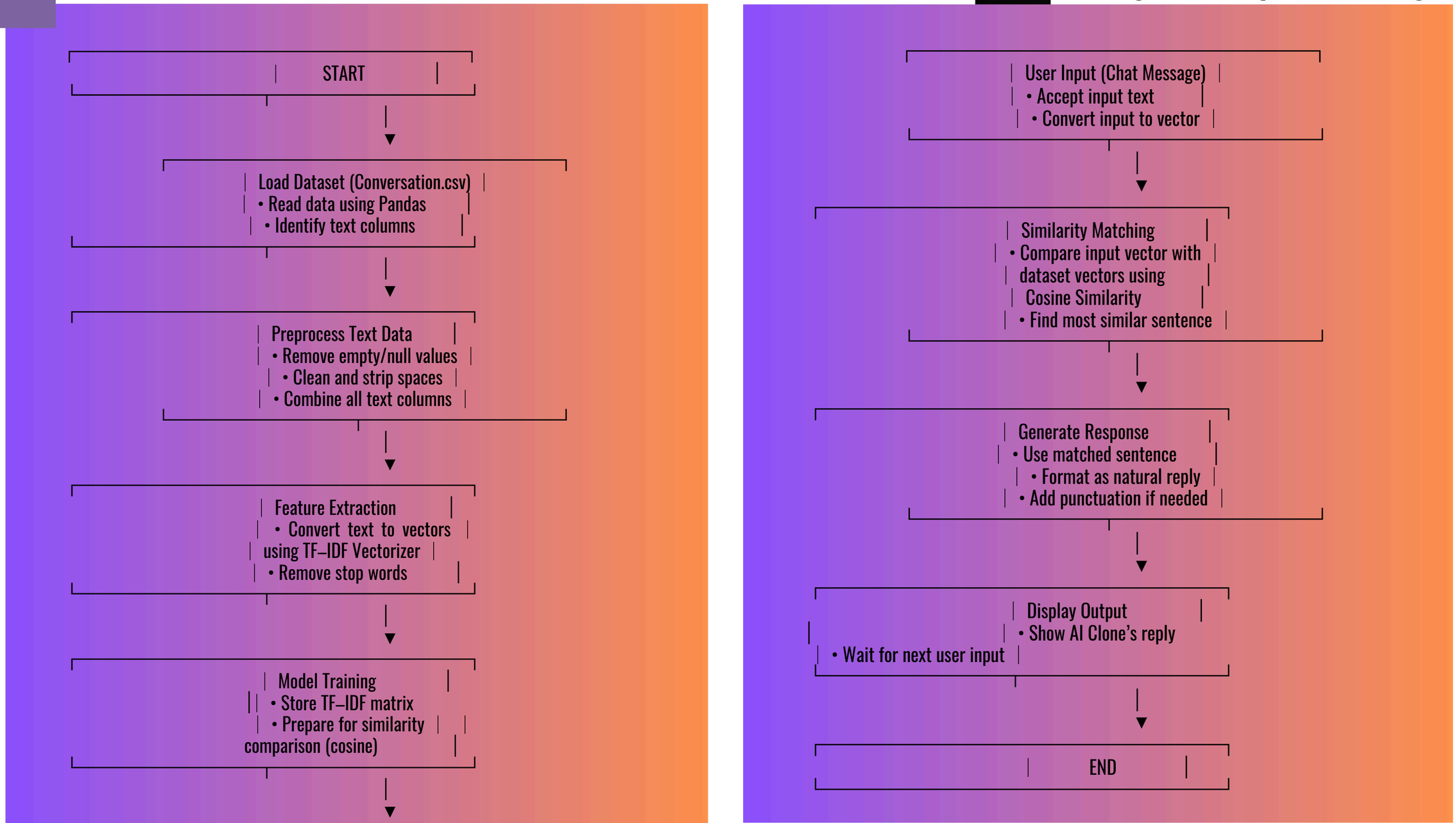
Methodology and Implementation Process

Step 1 – Data Collection
Load the user's Conversation.csv dataset.
Identify and extract all text columns for training.
Step 2 – Data Pre-processing
Remove null values, special characters, and blank lines.
Clean and normalize text to prepare for analysis.
Step 3 – Feature Extraction (TF–IDF)
Convert sentences into numerical feature vectors using TF–IDF.
Capture word frequency and importance across the dataset.
Step 4 – Model Building
Compute Cosine Similarity between user input and all dataset sentences.
Store the TF–IDF matrix for efficient real-time matching.
Step 5 – Response Generation
Retrieve the most similar sentence to the user's query.
Return it as the AI Clone's response with proper punctuation and tone.
Step 6 – Output Interface
Display responses in a chat-style console.
Continue conversation until user types exit.

JD

TECHNICAL APPROACH

TechSparks
CSE – Association
Department of Computer Science and Engineering



JD

FEASIBILITY AND VIABILITY

Technical Feasibility

Uses lightweight,open-source libraries (Python, Scikit-learn, Pandas).Can be easily implemented and run on a standard laptop — no GPU or cloud needed.

Requiresonly a .csv dataset andbasicPython environment (IDLE or Jupyter).Simple and explainable ML pipeline ensures low computational cost.

Operational Feasibility

Easy to operate through a console chat interface.Minimal user input — just provide the dataset and typemessages.Works entirely offline, protecting user data and ensuring privacy.Suitable for demonstration within 5-hour development time (hackathon-friendly).

Economic Feasibility

Zero cost implementation using open-source tools and local setup.

No licensing or server maintenance expenses.

Can scale to web or app versions later with minimal investment.

JD

IMPACT AND BENEFITS

Overall Impact

The Adaptive Digital Self–AIClonedemonstrates how Artificial Intelligence can ethically learn and mirror a person's communication style while ensuring privacy, personalization, and human-like interaction.
It bridges the gap between human expression and machine intelligence.

Technical Impact

Promotes the use of Explainable and Lightweight ML Models like TF–IDF and Cosine Similarity.

Encourages privacy-preserving AI systems that function completely offline.

Proves that personalized AI can be built without large datasets or deep neural networks.

Offers a modular foundation for advanced applications such as digital twins, writing assistants, and mental health bots.

Supports the development of context-aware chat systems using local data only.

User and Social Benefits

Helps usersunderstandand explore their own communication patterns through AI reflection.

Can serve as a personal digital companion for writing practice, journaling, or creative storytelling.

Ensures data safety — no external data sharing or online storage.

Promotes ethical AI usage and awareness of responsible data handling.

Provides accessible AI technology to students and researchers with minimal cost and setup.

Educational and Research Value

Demonstrates key Machine Learning and NLP concepts in an understandable way.

Useful for teaching AI ethics, personalization, and explainability.

Can be extended into capstone projects or academic research prototypes.

Future Benefits and Scalability

Can evolve into a multilingual AI companion capable of adapting tone and sentiment.

Potential integration with voice, emotion detection, and mental health monitoring.

Scalable to mobile or web-based chatbots for real-time interaction.

Encourages innovation in digital identity and self-reflective AI systems.

JD

REFERENCES


```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Step 1: Load dataset
file_path = r"E:\jd\Conversation.csv"  # path to your uploaded dataset
data = pd.read_csv(file_path, encoding="utf-8")

# Step 2: Automatically find text columns
text_columns = [col for col in data.columns if data[col].dtype == "object"]

if not text_columns:
raise ValueError("No text columns found in dataset!")

# Combine all text columns into one list
text_data = []
for col in text_columns:
text_data.extend(data[col].dropna().tolist())

# Step 3: Clean and prepare text data
text_data = [t.strip() for t in text_data if isinstance(t, str) and len(t.strip()) > 0]

if len(text_data) == 0:
raise ValueError("No usable text found in dataset!")

# Step 4: Create TF-IDF model
vectorizer = TfidfVectorizer(stop_words="english")
tfidf_matrix = vectorizer.fit_transform(text_data)

print(f"✅ Loaded {len(text_data)} sentences from your dataset.")
print("🤖 AI Clone ready! Type your message below (type 'exit' to quit)\n")

# Step 5: Generate AI responses (fixed version)
def generate_response(user_input):
# Convert user input into vector
user_vec = vectorizer.transform([user_input])

# Compute similarity with dataset text
similarities = cosine_similarity(user_vec, tfidf_matrix)
best_match_idx = similarities.argmax()
base_sentence = text_data[best_match_idx]

# ✅ FIX: Return full matched sentence instead of jumbled words
response = base_sentence.strip()

# Add punctuation if missing
if not response.endswith(('.', '!', '?')):
response += '.'

return response

# Step 6: Chat loop
while True:
user_input = input("You: ")
if user_input.lower() == "exit":
print("AI Clone: Goodbye! Keep exploring your digital self. 🌱")
break

reply = generate_response(user_input)
print("AI Clone:", reply)
```