

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELAGAVI - 590 018



A Major Project Phase 2 Report on
Virtual Universe Construction for Digital Afterlife using
Generative Neural Networks

A Project work submitted in the partial fulfillment for the award of the degree
Bachelor of Engineering
in
Artificial Intelligence & Machine Learning

Submitted by

Allene Aaisha	1AY22AI010
Britta Biju	1AY22AI021
Harshitha M V	1AY22AI038
Navyashree S	1AY22AI056

Under the Guidance of
Dr. Kavitha Nair R
Assistant Professor
Department of AI & ML



2025-2026

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
Acharya Institute of Technology

Acharya Dr. Sarvepalli Radhakrishnan Road, Soladevanahalli, Bengaluru-560107

ACHARYA INSTITUTE OF TECHNOLOGY
**Acharya Dr. Sarvepalli Radhakrishnan Road, Soladevanahalli,
Bengaluru 560107**

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING



CERTIFICATE

Certified that the Major Project Phase 2 entitled "**Virtual Universe Construction for Digital Afterlife using Generative Neural Networks**" is carried out by **Allene Aaisha (1AY22AI010)**, **Britta Biju (1AY22AI021)**, **Harshitha MV (1AY22AI038)** and **Navyashree S (1AY22AI056)** in the partial fulfillment for the award of the degree of Bachelor of Engineering in Artificial Intelligence & Machine Learning of Visvesvaraya Technological University, Belagavi during the year **2025-2026**. It is certified that all corrections/suggestions indicated for the assessment have been incorporated in the report deposited in the departmental library. The Major Project Phase 2 Report has been approved as it satisfies the academic requirement in respect of Major Project Phase 2 (BAI786) prescribed for the Bachelor of Engineering Degree.

.....
Signature of Guide
Dr. Kavitha Nair R
Assistant Professor
Dept of AI & ML, AIT

.....
Signature of HOD
Dr. Vijayashekhar S S
Head of the Department
Dept of AI & ML, AIT

.....
Signature of Principal
Dr. C K Marigowda
Principal, AIT

Name of the Examiners

Signature with date

1

2

DECLARATION

We the students of seventh semester Artificial Intelligence & Machine Learning, Acharya Institute of Technology, Soladevanhalli Bengaluru - 560107 declare that work entitled “VIRTUAL UNIVERSE CONSTRUCTION FOR DIGITAL AFTERLIFE USING GENERATIVE NEURAL NETWORKS” has been successfully completed under the guidance of Dr. Kavitha Nair R, Assistant Professor, Department of AI & ML, Acharya Institute of Technology, Bengaluru. This dissertation work is submitted to Visvesvaraya Technological University in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in Artificial Intelligence & Machine Learning during the academic year 2025-2026. Further the matter embodied in the Major Project Phase 2 report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Bengaluru

Date:

Allene Aaisha	1AY22AI010
Britta Biju	1AY22AI021
Harshitha M V	1AY22AI038
Navyashree S	1AY22AI056

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of a task would be incomplete without the mention of the people who made it possible and without their constant guidance and encouragement success would not have been possible.

We grateful to the institute **Acharya Institute of Technology** for its ideas and for having provided us with the good infrastructure and inspiring staff which has made this Project Phase 2 successfully.

We would like to express our sincere gratitude to **Dr. C. K. Marigowda**, Principal, AIT for all the facilities that has been extended by him throughout our work.

We heartily thank and express our sincere gratitude to **Dr. Vijayashekhar S S**, HOD, Dept. of AI & ML, AIT for his valuable support and a constant source of enthusiastic inspiration to steer us forward.

We would like to express our sincere gratitude to the Internal Guide **Dr. Kavitha Nair R**, Assistant Professor, Dept. of AI & ML, AIT for her invaluable guidance and support.

We would like to express our sincere gratitude to the Project Coordinators **Dr. Anupallavi S**, Assistant Professor, Dept. of AI & ML, AIT and **Prof. Sanjay P**, Assistant Professor, Dept. of CSE(DS), AIT for their valuable guidance and support.

Finally, we would like to express our sincere gratitude to our parents, all teaching and non-teaching faculty members, and friends for their moral support, encouragement, and help throughout the completion of the Project.

Allene Aaisha	1AY22AI010
Britta Biju	1AY22AI021
Harshitha M V	1AY22AI038
Navyashree S	1AY22AI056

ABSTRACT

The pursuit of a digital afterlife—where human consciousness, identity, and memories are preserved and perpetuated within a virtual environment—presents both a philosophical and technological frontier. This project explores the construction of a Virtual Universe as a habitat for digital personas using Generative Neural Networks (GNNs). These advanced models, including Generative Adversarial Networks (GANs) and Diffusion Models, enable the creation of realistic and dynamic environments, lifelike avatars, and emotionally intelligent agents capable of interacting with preserved digital identities.

The core idea is to synthesize a personalized, interactive, and evolving universe where the essence of a person—extracted from multimedia inputs such as voice recordings, video archives, social media history, writings, and biometric data—is reconstructed into a living digital presence. These personas can continue to grow, reflect, and interact within their unique digital realms, offering both continuity and evolution of self.

The system architecture includes modules for personality reconstruction, memory synthesis, environment generation, and emotional modelling. A key focus is placed on ensuring the authenticity of the emulated persona, the psychological impact of interactions, and the ethical considerations surrounding consent, legacy, and identity preservation.

TABLE OF CONTENTS

Chapter No	Title	Page No
1	INTRODUCTION	1
	1.1 Overview	1
	1.2 Existing System	1-2
	1.3 Objective	3
	1.4 Scope	4-6
2	PROBLEM STATEMENT	7
	2.1 Problem Statement	7
	2.2 Motivation	7-8
	2.3 Objectives	9
3	LITERATURE SURVEY	10
	3.1 Detailed Survey	10-12
	3.2 Summary	13-15
	3.3 Proposed Solution	16
4	SYSTEM REQUIREMENT SPECIFICATION	17
	4.1 Functional Requirements	17-18
	4.2 Non-functional Requirements	18-19

4.3	Hardware Requirements	19-20
4.4	Software Requirements	20-21
5	SYSTEM DESIGN	22
5.1	Data flow Diagram	22-25
5.2	Use Case Diagram	26-29
5.3	Activity Diagram	30-32
5.4	Class Diagram	33-35
5.5	ER Diagram	36-38
5.6	System Architecture	39-41
5.7	Module Design	42-45
6	IMPLEMENTATION	46
6.1	Algorithms / Methodologies	46-48
6.2	Tools and Technologies	49-51
7	SYSTEM TESTING	52-53
8	RESULTS AND DISCUSSIONS	54-57
9	CONCLUSION	58
	Appendices	59
	A. Sample Code	59-69
	B. Snapshots	70
	References	71

LIST OF FIGURES

5.1	Data Flow Diagram	22
5.2	Use Case Diagram	26
5.3	Activity Diagram	30
5.4	Class Diagram	33
5.5	ER Diagram	36
5.6	System Architecture	39
5.7	Module Diagram	42

LIST OF TABLES

3.1	Literature Survey Table	13-15
5.2	Scenarios Table	28-29
7.1	System Testing Test Cases	52-53
8.1	System Performance Evaluation Metrics	54

CHAPTER 1

INTRODUCTION

1.1 Overview

The rapid growth of artificial intelligence has changed how people interact with digital systems. Traditional AI applications focused on information retrieval and basic automation. However, recent advances in Generative Neural Networks (GNNs), Retrieval-Augmented Learning, Voice Synthesis, and Talking-Head Generation allow for the creation of realistic digital identities. These identities can generate human-like responses, use cloned voices, and even appear in videos with synchronized lip movements. The idea of a Virtual Universe for Digital Afterlife takes these technologies further. It aims to preserve and recreate a person's knowledge, appearance, voice, and personality, allowing them to keep "living" digitally after their physical presence is gone. By combining multiple AI modules—text retrieval, natural language generation, voice cloning, and video animation—this project builds a system that can simulate conversations in a realistic and interactive way. It reconstructs the "digital persona" of a specific individual (Ratan Tata) to show how future generations can engage with preserved knowledge and identity. This chapter presents the foundation of the project, the need to preserve digital identity, existing limitations, and the goals and scope of the proposed solution.

1.2 Existing System

Before the rise of multimodal AI systems, most digital interactions were limited to:

1.2.1 Chatbots

Rule-based or intent-based chatbots rely on predefined answers.

Limitations:

- Responses are repetitive
- No personalization
- Cannot reconstruct a real person's personality

1.2.2 Voice Assistants

Tools like Siri or Google Assistant generate generic voice responses, but they cannot imitate the voice of a specific person.

Limitations:

- No voice cloning
- No emotional tone
- Not connected to personal knowledge

1.2.3 Deepfake or Talking-Head Videos

Current tools can animate a person's face but are usually:

- Pre-generated
- Not interactive
- Not connected to real knowledge or real-time responses

1.2.4 Memory Preservation Methods

Biographies, journals, videos, or speeches can keep a person's work, but:

- They are static
- Cannot answer new questions
- Cannot interact

GAP in Existing Systems

There is no fully integrated system that can:

- Understand user queries
- Retrieve real knowledge
- Generate a contextual answer
- Speak it using a cloned voice
- Produce a talking-head lip-synced video
- All in real-time

This gap motivates the development of the proposed system.

1.3 Objective

1.3.1 Develop a multimodal AI system that processes text, audio, and video for realistic digital interactions

The main goal of this project is to create a fully integrated multimodal artificial intelligence system that can understand and generate text, audio, and video in one process. Unlike traditional chatbots that only use text, this system combines natural language processing, voice synthesis, and talking-head video creation to mimic real human interactions. The model uses semantic retrieval for accuracy, an offline LLM to produce natural language responses, a neural TTS model to create cloned speech, and a generative video model to animate lip movements and facial expressions.

1.3.2 Create an interactive digital persona that mimics natural speech, expressions, and knowledge using AI-generated voice, video, and text

The second objective focuses on building a digital persona that can imitate the knowledge, tone, and behaviour of a real human. This involves capturing the individual's speaking style, voice patterns, communication habits, and domain knowledge through large transcript datasets and voice samples. The system is designed to answer questions contextually using information it retrieves. Through this objective, the digital persona becomes more than a static avatar. It turns into a dynamic AI entity that can engage in conversations, express emotions through synthesized voice, and show natural-looking facial movements.

1.3.3 Build an immersive virtual universe and make sure to keep long-term access through secure cloud deployment of the AI system

The third objective is to expand the digital persona into an immersive virtual universe, where the recreated identity can exist, interact, and provide knowledge for future generations. This includes hosting the system on scalable cloud infrastructure to guarantee long-term accessibility, data security, and reliable performance. Cloud deployment makes the digital avatar accessible from anywhere, preserves all multimodal data (text, audio, video), and keeps it functional even as hardware systems change. This objective ensures that the digital afterlife system is not only technologically sound but also stable, future-proof, and available as a lasting digital legacy.

1.4 Scope

The scope of this project covers multimodal artificial intelligence, digital identity reconstruction, and virtual human-computer interaction. The system aims to create an intelligent digital persona that can produce text, voice, and video responses similar to the knowledge, communication style, and facial expressions of a real person. This project shows how generative neural networks can help preserve a person's legacy through an interactive digital afterlife system.

1.4.1 Multimodal Interaction Capability

The project focuses on designing and implementing a multimodal AI framework. It can understand and create text, produce high-quality speech, and generate lip-synchronized video outputs. This enhances the interaction experience beyond traditional chatbots and offers a realistic, human-like digital presence. The scope includes natural language processing, text-to-speech conversion, and generative video synthesis. This ensures the system responds smoothly in all three modalities.

1.4.2 Knowledge Preservation and Retrieval

A major part of the project is preserving specific knowledge through a custom dataset that includes interviews, speeches, quotes, and biographical text. The system uses semantic embedding models and FAISS similarity search to find relevant information. This enables the digital persona to answer questions correctly based on the preserved knowledge, making the avatar both informative and consistent with the real individual's views.

1.4.3 Persona Reconstruction Using Artificial Intelligence

The scope includes reconstructing the communication style, tone, and personality traits of the selected individual. This involves modeling their speaking style with voice cloning, maintaining their response style through contextual language generation, and showing natural facial movements with talking-head video synthesis. The project makes sure that the recreated persona acts in a way that matches the original individual's identity.

1.4.4 Voice Cloning and Speech Synthesis

A key aspect of the project is using neural text-to-speech models to replicate a specific

voice. The system includes voice embedding extraction, speech generation, and prosody modeling to create a realistic audio output. This allows the digital persona to speak in a voice that closely matches the target individual, improving authenticity and emotional impact.

1.4.5 Talking-Head Video Generation

The project uses generative video models to animate a static image and create lip-synchronized video responses. This involves facial landmark detection, expression synthesis, and synchronized lip movement generation. The final product is a talking-head video that visually represents the digital persona. It offers an immersive and interactive way to communicate.

1.4.6 Real-Time Conversational Pipeline

The integrated system handles user queries, retrieves relevant data, generates appropriate responses, and produces multimodal outputs through a sequential processing pipeline. Video processing is demanding on CPU hardware, but the system still maintains full functionality and serves as a working prototype for real-time conversational avatars.

1.4.7 Digital Afterlife and Legacy Preservation

The project helps develop digital afterlife technologies by allowing long-term preservation of a person's knowledge, speech, and visual identity. The created digital persona can be used for education, remembrance, digital museums, and interactive biographical experiences. This broadens the focus from technical testing to meaningful social and emotional uses.

1.4.8 Cloud Deployment and Future Accessibility

To ensure long-term accessibility, the scope includes the potential deployment of the system on cloud platforms. This allows multiple users to interact with the digital persona remotely, ensures scalability, and enables ongoing improvement of the model through updated data.

1.4.9 Modular Design for Scalability and Extension

The system uses a modular architecture. This design allows for the addition of new models or features in the future. Possible extensions include 3D avatar generation, emotional speech synthesis, gesture animation, memory-enhanced language models, and real-time processing with GPU acceleration.

1.4.10 Limitations Within Scope

While the project achieves end-to-end multimodal generation, certain limitations define the current scope:

- Video generation is slower on CPU hardware
- Voice quality depends on reference audio clarity
- Emotional expressiveness is limited
- The avatar does not generate full-body animations
- Ethical considerations restrict certain use-cases

These limitations outline what is included and excluded from the current version of the project.

CHAPTER-2

PROBLEM STATEMENT

2.1 Problem Statement

Human identity is a mix of knowledge, memories, personality traits, communication style, emotional expression, and life experiences. When someone dies, all these aspects are permanently lost, leaving behind only static remnants like photographs, books, interviews, or audio recordings. These traditional ways of preserving memory cannot capture the lively, interactive nature of a person. They cannot answer new questions, adjust their responses, or engage in natural conversations.

Despite major improvements in artificial intelligence, most existing digital systems focus on text-only chatbots, generic voice assistants, or pre-made deepfake videos. These tools cannot truly represent a real person, nor can they keep a person's knowledge organized in an interactive way. There is also no single platform that brings together text understanding, LLM-generated responses, voice cloning, and talking-head video generation into one system.

This brings us to the main issue:

How can we create an AI-based system that keeps the knowledge, voice, personality, and appearance of an individual, allowing future generations to interact with their digital representation in a realistic, meaningful, and multimodal way?

The goal is to develop a digital persona that not only generates text responses but can also speak in a cloned voice and produce lip-synced videos, making conversations feel human-like. Additionally, the system must ensure factual accuracy, reduce errors, keep identity consistent, and work efficiently even with limited hardware.

Therefore, the main issue this project addresses is the lack of an integrated, multimodal, retrieval-augmented generative AI system that can create a digital afterlife avatar capable of existing in a virtual universe.

2.2 Motivation

The motivation for this project comes from personal and technological inspirations. On a personal level, the wish to digitally recreate and preserve someone meaningful, like Ratan

Tata, an iconic leader known for his values and contributions to society, shows the emotional drive behind the project. Many people have heroes, family members, or loved ones they wish they could still talk to, learn from, or hear. The concept of a digital afterlife offers a way to keep these connections alive even when someone is no longer physically present.

On a larger scale, society is producing vast amounts of digital content, but we still lack systems that turn this information into interactive digital personas. The growth of generative neural networks, retrieval-augmented learning, and high-quality voice and video technologies gives us a unique chance to create systems that can preserve human legacies in ways that were not possible before.

Key motivations include:

2.2.1 Preserving Knowledge and Legacy

Leaders, innovators, scientists, and cultural icons have valuable wisdom. Creating digital avatars of these individuals ensures that their experiences and teachings remain accessible to future generations.

2.2.2 Emotional and Psychological Value

Digital personas can help people reconnect with loved ones or mentors. They offer comfort, continuity, and a sense of presence, especially for those grieving a loss.

2.2.3 Educational and Historical Applications

Students can engage with digital versions of historical figures, gaining insights in a more personal and conversational way.

2.2.4 Technological Innovation

Developments in generative AI present exciting opportunities for new ways that humans and machines can interact.

2.2.5 Personal Dream and Inspiration

In your case, not being able to meet Ratan Tata in person drives you to create a digital representation that preserves his thoughts, voice, and presence for future generations.

This emotion-driven motivation enhances the project's authenticity and value.

2.3 Objectives

The main goal of the project is to create a multimodal digital avatar system that can simulate text, voice, and video responses like a real person. To do this, the project uses generative neural networks and retrieval-augmented models to build an interactive digital afterlife environment.

The objectives are as follows:

2.3.1 Develop a multimodal AI system that processes text, audio, and video for realistic digital interactions.

This involves combining NLP, TTS, and video generation models into one working pipeline. The system must understand user questions, find relevant information, generate suitable answers, speak them using a cloned voice, and create videos with precise lip synchronization.

2.3.2 Create an interactive digital persona that imitates natural speech, expressions, and knowledge using AI-generated voice, video, and text.

The digital persona must act consistently with the communication style of the real person. This includes generating answers that reflect the person's values, speaking with the same tone and voice quality, and animating a facial image with lifelike expressions.

2.3.3 Build an immersive virtual universe and ensure long-term access through secure cloud deployment of the AI system.

The aim is to create a platform where the digital persona can exist without physical limits. The system should be available at all times and should allow future integration of 3D avatars, emotional modeling, and memory-based personalization. Cloud deployment guarantees scalability and long-term preservation of the digital identity.

CHAPTER-3

LITERATURE SURVEY

3.1 Detailed Survey

1. Neural Voice Puppetry: Audio-driven Facial Reenactment (2019)

Authors: Justus Thies, Mohamed Elgharib, Ayush Tewari, Christian Theobalt, Matthias Nießner

This pioneering work introduces a technique for generating photorealistic facial reenactment videos driven entirely by audio input. The model captures facial expressions and synchronizes them with voice, laying the foundation for realistic talking avatars in digital afterlife applications.

2. Audio-driven Talking Face Video Generation with Learning-based Personalized Head Pose (2020)

Authors: Ran Yi, Zipeng Ye, Juyong Zhang, Hujun Bao, Yong-Jin Liu

This paper presents a model for producing natural head movements in talking face videos using personalized learning-based head pose prediction, increasing the realism of avatars.

3. A Survey of Audio Synthesis and Lip-syncing for Synthetic Video Generation (2021)

Authors: Anup Kadam, Sagar Rane, Arpit Kumar Mishra, Shailesh Kumar Sahu, Shubham Singh, Shivam Kumar Pathak

An extensive survey covering state-of-the-art tools like WaveNet and Wav2Lip. It explores applications, challenges, and innovations in audio synthesis and video lip-syncing.

4. LipSync3D: Data-Efficient Learning of Personalized 3D Talking Faces from Video (2021)

Authors: Avisek Lahiri, Vivek Kwatra, Christian Frueh, John Lewis, Chris Bregler
Introduces LipSync3D, a framework that animates 3D faces from audio while maintaining identity-specific visual features, essential for personal digital avatar realism.

5. Imitator: Personalized Speech-driven 3D Facial Animation (2022)

Authors: Balamurugan Thambiraja, Ikhsanul Habibie, Sadegh Aliakbarian, Darren Cosker, Christian Theobalt, Justus Thies

This paper proposes a method for generating 3D facial animations personalized to the subject's identity and speaking style, enhancing emotional realism in avatars.

6. SyncTalkFace: Talking Face Generation with Precise Lip-Syncing via Audio-Lip Memory(2022)

Authors: Se Jin Park, Minsu Kim, Joanna Hong, Jeongsoo Choi, Yong Man Ro
Proposes an audio-lip memory mechanism to improve lip-sync precision in avatar videos.
This helps ensure natural facial expressions and speech synchronization.

7. Multimodal Multilingual Avatar Video Generation in Digital Marketing (2024)

Describes an AI system that generates customizable avatar videos from text, audio, or images. Multilingual and multimodal input support makes it applicable to afterlife simulations across cultures.

8. The Afterlife of Data: Who Controls Our Digital Legacy? (2024)

Author: Caroline Carruthers

Explores digital legacy management, emphasizing the legal and ethical issues around posthumous data ownership, which is vital for AI-based afterlife systems.

9. Generative Ghosts: Anticipating Benefits and Risks of AI Afterlives (2024)

This research explores how AI can simulate deceased individuals, raising questions around identity, consent, and the long-term psychological impact of such simulations.

10. The Law of Digital Resurrection (2025)

Examines the legal frameworks needed to govern AI-driven digital resurrection of people.
It discusses digital personhood, consent, and posthumous rights.

11. Seeing the Sound: Multilingual Lip Sync for Real-Time Face-to-Face Translation (2025)

Authors: Amirkia Rafiei Oskooei, Mehmet S. Aktaş, Mustafa Keleş

This paper evaluates real-time multilingual lip-sync performance, enabling avatars to communicate naturally in various languages with accurate audio-visual alignment.

12. AI Afterlife as Digital Legacy: Perceptions, Expectations, and Ethical Considerations (2025)

Explores how different cultures and user groups perceive AI-powered digital afterlife systems, with a focus on public trust, ethical boundaries, and emotional impact.

13. AI and the Afterlife: A Psychological, Ethical, and Technological Analysis (2025)

Author: Fabrizio Degni

This comprehensive study analyzes psychological, ethical, and technical challenges in simulating digital consciousness and resurrecting personalities using generative models.

14. Learning through AI-clones: Enhancing Self-perception and Communication Skills (2025)

Investigates the psychological benefits of interacting with AI-based clones, such as enhancing self-awareness and communication abilities, especially useful in legacy systems.

15. The Digital Afterlife: AI Cloud Consciousness as the New Immortality (2025)

Discusses the theoretical and technological underpinnings of “cloud consciousness,” where human memory, personality, and behavior are replicated in AI form for digital immortal

3.2 Survey Summary Table

Table 3.2 Survey Summary Table

Sl. No	Title of the Paper and Journal	Problem Addressed	Authors Approach	Results
1	AI-Generated “Generative Ghosts” – <i>Nature Machine Intelligence</i> (2024)	Simulation of deceased individuals using digital avatars	Deep generative avatar models with conversational AI	Achieved lifelike avatar interaction but raised major ethical concerns and showed limited realism
2	DreamAvatar: Text-to-Avatar Generation via 3D Diffusion Models (2024)	Automatic generation of full-body avatars from text descriptions	3D diffusion model conditioned on textual prompts and skeletal constraints	Generated customizable avatars from text but suffered from slow generation time
3	EMO: Emote Portrait Alive (2024)	Real-time emotional facial animation from audio	Audio-driven neural facial animation model with emotion control	Produced realistic emotional expressions but struggled with generalizing to new identities
4	Ethical Challenges of AI-Driven Avatars – <i>AI</i> (2023)	Ethical risks in avatar recreation and identity cloning	Multidisciplinary socio-technical survey and policy review	Identified privacy and consent risks without concrete mitigation frameworks

5	XTTS v2: Multilingual Cross-Speaker TTS – <i>Coqui AI Report</i> (2023)	Multilingual voice cloning and speech synthesis	Cross-speaker transformer-based TTS pipeline	Delivered high-quality voice transfer but with limited emotion range and latency
6	RVC: Real-Time Voice Cloning – <i>Github Project</i> (2023)	Fast personalized speech synthesis	Lightweight deep learning audio cloning pipeline	Achieved quick voice mimicry with acceptable quality using large datasets
7	Simsensei – Virtual Human Interaction System (USC ICT Study) (2023)	Creating emotionally responsive virtual humans for mental health conversations	Multimodal affect- sensing combined with dialogue and gesture generation	Produced realistic empathic responses but limited by scripted emotional range
8	VALL-E: Neural Codec Language Model for TTS – <i>Microsoft Research</i> (2023)	Zero-shot voice cloning with minimal samples	Neural codec language modeling using discrete acoustic tokens	Demonstrated accurate voice cloning from 3- second samples but limited emotional variability
9	PanoHead: 3D Full-Head Synthesis – <i>CVPR</i> (2023)	Generating high-resolution animatable 3D digital heads	Multi-view GAN pipeline producing 3D-consistent head geometry and textures	Achieved high-quality 3D heads but required multi-angle face data

10	ChatGPT: Dialogue-Based LLM Interaction – <i>OpenAI Technical Report</i>	Natural conversational agents for dialogue systems	Large transformer-based generative pre-trained language model	Produced fluent dialogue but lacked long-term memory consistency
11	Make-A-Video: Text-to-Video Generation – Meta AI	Generating short, realistic videos from text prompts	Diffusion-based text-to-video model using frozen language encoders	Generated visually coherent videos but struggled with temporal consistency
12	FaceFormer: Speech-Driven 3D Facial Animation – <i>CVPR</i>	Speech-to-facial expression synchronization	Transformer models mapping audio features to 3D face meshes	Delivered realistic lip-sync but required extensive training data
13	D-ID Creative Reality Studio – <i>White Paper</i>	Automated talking-head video synthesis	GAN-based face animation driven by audio and text	Generated photorealistic talking avatars with limited expressiveness
14	StyleGAN3: Alias-Free GANs – <i>NeurIPS</i>	Creation of cinematic-quality real-time digital avatars	Creation of cinematic-quality real-time digital avatars	Generated highly realistic avatars but without embedded AI interaction
15	MetaHumans: High-Fidelity Digital Humans – <i>Epic Games Report</i>	Creation of cinematic-quality real-time digital avatars	Scanned-based avatar creator integrated with Unreal Engine	Generated highly realistic avatars but without embedded AI interaction

3.3 Proposed Solution

The proposed solution introduces an AI-driven system designed to create a lifelike digital avatar. This avatar can interact with users through text, voice, and video outputs. The system uses Retrieval-Augmented Generation (RAG) to extract relevant information from large transcripts. This method ensures that the answers are factual and context-driven. An LLM processes these extracted chunks to produce fluent responses that align with the avatar's personality.

To improve the realism of the avatar, the system employs XTTS for high-quality voice synthesis. This enables accurate voice cloning from a reference speech sample. Additionally, SadTalker generates lip-synced video animations of the avatar. This delivers a multimodal response that closely resembles the real person's appearance and speaking style.

By bringing these components together, the solution addresses the limitations in the current research. It connects knowledge retrieval, conversational modeling, voice identity, and visual realism. The system can serve as a foundation for digital afterlife experiences, interactive memorials, education, and historical preservation. It ensures that an individual's knowledge and persona can be digitally preserved for future generations.

CHAPTER-4

SYSTEM REQUIREMENT SPECIFICATION

The System Requirement Specification (SRS) outlines all the functional, non-functional, hardware, and software requirements needed to design, develop, test, and deploy the proposed Virtual Universe Construction for the Digital Afterlife system. The SRS provides clarity, consistency, and completeness about what the system needs to accomplish and how it should function in different situations.

4.1 Functional Requirements

Functional requirements describe what the system must do. They define the main operations, behaviours, and features that allow the system to create text, audio, and video output as part of the multimodal AI pipeline.

4.1.1 User Query Handling

- The system shall accept user questions in text format.
- The system shall validate queries to ensure they relate to the chosen persona, Ratan Tata.
- The system shall provide alerts for invalid or unrelated queries.

4.1.2 Data Retrieval Using Semantic Search

- The system shall use the MiniLM embedding model to convert user queries into vector embeddings.
- The system shall retrieve the most relevant transcript chunks with FAISS similarity search.
- The system shall combine the top-k retrieved chunks into a context block.

4.1.3 Text Response Generation (LLM Output)

- The system shall generate a text response using GPT4All or similar lightweight LLM models.
- The system shall use the retrieved context to avoid hallucination and ensure factual accuracy.

- The system shall provide concise and meaningful responses to the user question.

4.1.4 Voice Generation (Text-to-Speech)

- The system shall convert the generated text response into speech using XTTs v2.
- The system shall clone the target individual's voice using a reference audio file.
- The output shall be saved as a WAV audio file.

4.1.5 Video Generation (Talking-Head Animation)

- The system shall create a lip-synced video using SadTalker.
- The system shall animate a static image to match the generated audio.
- The system shall save the output in MP4 format.

4.1.6 Output Delivery

- The system shall return three outputs: text, audio, and video.
- The system shall preview the generated video within the interface.
- The system shall allow users to save or view the generated multimodal responses.

4.1.7 Error Handling

- The system shall notify the user when input is invalid.
- The system shall handle missing files, corrupted audio, or long-duration failures smoothly.
- The system shall provide logs to help debug errors related to model inference.

4.2 Non-functional Requirements

Non-functional requirements define system performance standards, design limits, usability, reliability, and other quality features.

4.2.1 Performance Requirements

- Text response generation should finish within 20 to 30 seconds on a CPU.
- Voice generation should finish within 40 to 60 seconds.
- Video generation may take 2 to 5 minutes, depending on hardware capability.
- Semantic retrieval must work within 1 to 2 seconds.

4.2.2 Accuracy Requirements

- Knowledge retrieval accuracy should be over 85 to 90%.
- Voice similarity score should be 90% or above based on reference audio.
- Video lip-sync alignment should look realistic and be synchronized.

4.2.3 Reliability Requirements

- The system must work consistently for all valid inputs.
- All models should load and run without interruption.
- Audio and video output must be produced even under CPU limits.

4.2.4 Usability Requirements

- The system interface should be simple and easy for non-technical users.
- Users should only need to type a question to get responses.
- Instructions and error messages must be clear and easy to understand.

4.2.5 Scalability Requirements

- The system must allow adding new transcripts, voices, or persona data.
- The setup must support cloud deployment.
- Future upgrades to 3D avatars, emotions, and memory features must be possible.

4.2.6 Security Requirements

- All text and audio data must be stored securely.
- Access to voice cloning models must be controlled to prevent misuse.
- The system must implement permissions for future multi-user access.

4.2.7 Maintainability Requirements

- All modules must follow modular coding standards.
- Data pipelines should be easy to adjust.
- The codebase should support upgrading models without complete redesign.

4.3 Hardware Requirements

These requirements outline the minimum hardware needed to run the multimodal AI pipeline effectively.

4.3.1 Minimum Hardware Configuration

- Processor: Intel Core i5, Ryzen 5, or higher
- RAM: 12 GB
- Storage: 20 GB free space
- Graphics Card: Not mandatory; CPU-only models can be used
- Audio Device: Basic sound output for audio preview

4.3.2 Recommended Hardware Configuration

- Processor: Intel Core i7, Ryzen 7
- RAM: 16 to 32 GB
- SSD Storage: 50 GB free
- Dedicated GPU: NVIDIA GTX 1650, RTX series for faster video generation
- High-speed internet is required if cloud integration is used
- A GPU is not mandatory, but SadTalker runs much faster with GPU acceleration.

4.4 Software Requirements

Software requirements define all the necessary operating systems, libraries, frameworks, and development tools used in the project.

4.4.1 Operating System

- Windows 10, Windows 11
- Linux (Ubuntu 20.04+) is optional.
- macOS is compatible but not recommended for SadTalker GPU mode.

4.4.2 Programming Languages

- Python 3.10 or higher.

4.4.3 Important Libraries & Frameworks

- FAISS is for vector similarity search.
- SentenceTransformers provides MiniLM embeddings.
- GPT4All is an offline LLM.
- XTTs v2 (Coqui TTS) is for voice cloning.

- SadTalker handles video generation.
- PyTorch is used for model inference.
- NumPy, SciPy, and scikit-learn are for computations.
- MoviePy, OpenCV, and PIL are for video and image handling.

4.4.4 Development & Deployment Tools

- Python virtual environment
- VS Code, PyCharm
- FFmpeg
- Git for version control
- Optional cloud services include AWS, GCP, or Azure

CHAPTER-5

SYSTEM DESIGN

5.1 Data Flow Diagram

5.1.1 Purpose of the DFD

The Data Flow Diagram (DFD) shows how information moves through the system. It illustrates the flow of data from the user to the Retrieval Module, then to the LLM, and finally to the voice and video generation modules. The DFD helps to understand system behavior, dependencies, and how information moves. The architecture of the "Virtual Universe" system is divided into four core modules:

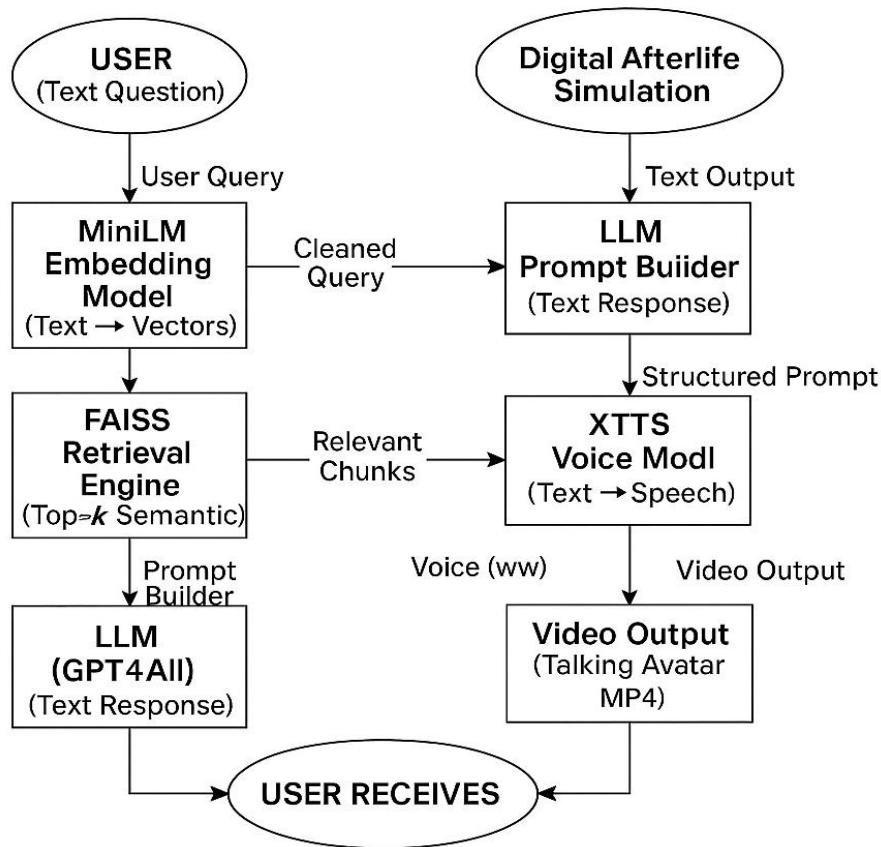


Fig 5.1 Data Flow Diagram of Virtual Universe Digital Afterlife System

This diagram aims to clarify:

- How user queries are processed
- How knowledge retrieval takes place
- How AI-generated text turns into speech
- How speech becomes a realistic talking video avatar

5.1.1 Purpose of the Data Flow Diagram

The main goal of this DFD is to:

- Show the overall data movement in the system
- Illustrate how retrieval, generation, and rendering modules connect
- Help understand the dependencies between AI components
- Provide transparency in multimodal pipeline processing

The DFD confirms that the system uses a Retrieval-Augmented Generation (RAG) architecture along with speech and video synthesis pipelines.

5.1.2 Detailed Explanation of the Data Flow

a) User (Text Question Input)

The process starts with the User, who submits a question through the chatbot interface. This question serves as the main input that triggers the entire AI pipeline.

This input may include:

- Informational questions
- Personal curiosity questions
- Knowledge-based queries

b) MiniLM Embedding Model (Text → Vectors)

After receiving the question, it is sent to the MiniLM Embedding Model.

MiniLM transforms the user's question into 384-dimensional numerical vectors.

These embeddings capture the meaning of the question rather than just relying on keywords.

This step ensures:

- Clear semantic understanding
- Better retrieval accuracy
- Effective handling of rephrased or indirect queries

c) FAISS Retrieval Engine (Vector → Knowledge Chunks)

The MiniLM vector goes to the FAISS Retrieval Engine, which:

- Stores embeddings of over 11,000 transcript chunks
- Conducts Top-K semantic similarity searches
- Retrieves the most relevant transcript sections

Benefits include:

- No hallucination
- Fact-based response generation
- Improved knowledge reliability

d) Prompt Builder (Context Structuring)

The retrieved knowledge chunks go to the Prompt Builder, which combines:

- User query
- Retrieved transcript context
- System personality instructions

Output:

A structured prompt that guides the LLM in generating accurate and personality-aligned answers.

e) LLM – GPT4All (Text Response Generation)

The structured prompt is sent to GPT4All, which:

- Processes the retrieved information
- Generates a contextual textual answer
- Maintains the style and tone of the selected persona

Output:

A high-quality AI-generated text response.

f) XTT Voice Cloning (Text → Speech)

The generated text response is passed to the XTT Voice Cloning Model, which:

- Converts the text into speech
- Uses reference audio samples
- Produces realistic human-like voice output in WAV format

This provides:

- Natural speech
- Emotional expressiveness
- Identity-specific voice recreation

g) SadTalker Video Generation (Speech → Lip-synced Video)

The generated audio and reference image go to SadTalker, which:

- Creates facial animation
- Applies lip synchronization
- Produces a talking avatar video in MP4 format

The video:

- Matches the timing of the speech
- Mimics natural mouth movement
- Offers a fully immersive digital avatar experience

h) Final Output Delivered to User (Multimodal Response)

The final outputs return to the User Interface, including:

- Text Answer
- Voice Output
- Talking Avatar Video

The user can:

- Read the response
- Listen to the voice
- Watch the avatar speak

5.1.3 Key Strengths of the Data Flow Model

- Ensures low hallucination by using FAISS-based retrieval.
- Maintains high semantic accuracy with MiniLM.
- Provides realistic voice cloning through XTTS.
- Produces high-quality lip-synced avatar videos with SadTalker.

5.2 Use Case Diagram

The use case diagram shows all the interactions between the user and the Virtual Universe Digital Afterlife System. It helps to identify system boundaries, actors, and the main functions that the system offers. The diagram illustrates how a user triggers different multimodal outputs like text, audio, and video.

5.2.1 Use Case Diagram

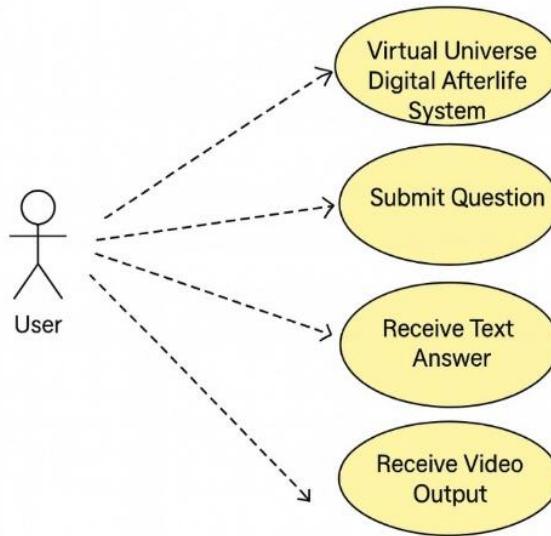


Fig 5.2 UML Use Case Diagram

5.2.2 Actor Description

Primary Actor: User

The User is the main external actor who interacts with the Virtual Universe system. The user:

- Enters text-based questions through a simple interface.
- Requests output in multiple formats, including text, audio, and video.
- Does not need any technical or programming knowledge.
- Interacts through a user-friendly chatbot-style interface.
- The system is built to be easy for all users, including those who are not technical, making it simple to use and intuitive to navigate.

5.2.3 System Description

The system works as a multimodal digital avatar that:

- Accepts text input
- Retrieves relevant knowledge
- Generates text, voice, and video responses
- Delivers all outputs to the user

5.2.4 Use Cases

Use Case 1: Submit Question

- Description: The user enters a query in text format.
- Goal: To request information from the digital persona.
- Pre-condition: The system must be active.
- Post-condition: The query is processed by the retrieval module.

Use Case 2: Receive Text Answer

- Description: The system provides a text response generated using GPT4All and retrieved context.
- Goal: Deliver a context-aware and accurate answer.
- Pre-condition: The retrieval and LLM modules must function.
- Post-condition: The answer is displayed on the screen.

Use Case 3: Receive Voice Output

- Description: The system converts the generated text into cloned voice using XTTs v2.
- Goal: Provide natural-sounding speech resembling the target persona.
- Pre-condition: The audio generation pipeline must be active.
- Post-condition: A WAV file is delivered.

Use Case 4: Receive Video Output

- Description: The system generates a lip-synced video using SadTalker.
- Goal: Provide a realistic talking avatar output.
- Pre-condition: SadTalker models and image must load successfully.
- Post-condition: An MP4 video is delivered.

5.2.5 Scenarios

Table 5.2 Scenarios of usecase

Scenario No.	Title	Actor(s)	Objective	Steps Involved	Expected Outcome
1	Avatar Creation from User Input	End User	Create a personalized avatar using voice and image data	1. Log into the system 2. Upload voice recordings and facial images 3. System validates and preprocesses input 4. Avatar is generated	User receives a realistic avatar generated using StyleGAN and voice cloned via XTTs v2
2	Lip-Synced Video Generation	End User	Generate a talking avatar video synced with the user's voice	1. Input or upload text 2. Generate speech using cloned voice 3. Avatar is lip-synced using Wav2Lip 4. Preview and download video	User obtains a personalized video where the avatar speaks with their cloned voice

Scenario No.	Title	Actor(s)	Objective	Steps Involved	Expected Outcome
3	Research and Model Benchmarking	Developer/ Researcher	Test and evaluate new generative models or pipelines	1. Integrate new model (e.g., GAN variant) 2. Run same input through both pipelines 3. Log metrics and compare quality	Developers gain insights into model performance and can iteratively improve the pipeline
4	Report and Output Export	End User, Researcher	Share results or preserve avatar outputs	1. Choose output (avatar image, video, report) 2. Click “Export” 3. Select format (PDF/MP4/HTML) 4. Download or email	Final avatar, video, and summary report are exported for archival or sharing
5	Consent and Data Management	End User	Provide data usage consent and manage personal data	1. Review terms and storage policy 2. Select consent options 3. Submit 4. Optionally delete or anonymize previous uploads	System stores data based on user consent or deletes it per user instructions

5.3 Activity Diagram

The Activity Diagram shows the operational workflow of the system, detailing how user input changes into multimodal output through a series of steps. This diagram is important for understanding how control flows within the system, the decision points involved, and the generation of data at the same time.

The activity diagram for the “Virtual Universe Construction for Digital Afterlife” project illustrates the entire process, from receiving a user query to producing text, voice, and video responses. It highlights both the sequential and concurrent actions that lead to the multimodal output.

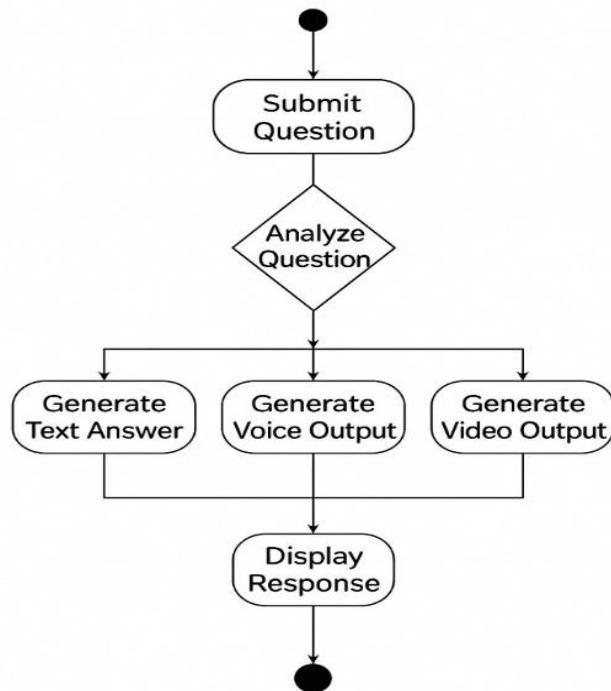


Fig 5.3 Process Flow Activity Diagram

5.3.1 Overview

The activity diagram shows the full execution flow of the system from:

User Input, Processing, Retrieval, Text Generation, Voice Synthesis, Video Synthesis, Output Delivery

5.3.2 Workflow Explanation

a) Start of the Process

- The process begins when the system launches and the user starts an interaction. The start node, a filled black circle, shows that the workflow has been activated.
- At this point, the system is idle, waiting for user input.

b) User Enters Question

The first major activity is the user submitting a question. This is the main input for the system and triggers all subsequent processes.

Characteristics:

- The input must be in text form.
- The question should relate to the target individual, Ratan Tata.
- The input module captures the text entry.
- This activity begins the multimodal AI pipeline.

c) Input Validation and Relevance Check

Once the question is submitted, the system checks for validation:

The system verifies:

- Whether the question is empty.
- Whether it contains meaningful content.
- Whether it is relevant to the persona, for example, contains "Ratan Tata," "Tata Group," etc.
- Whether it follows the expected format.

Decision Node:

If the question is not relevant, the system:

- Displays an error message.
- Ends the workflow.

If the question is relevant, the workflow moves to the next stage. This keeps the system focused and avoids irrelevant responses.

d) Embedding Generation (MiniLM)

After validation, the question goes into the Embedding Module, where:

- The text is turned into a numerical vector, known as an embedding.

- The embedding reflects meaning and semantic content.
- The MiniLM model is used for quick and accurate embedding generation.

This embedding helps the system to perform semantic similarity matching, ensuring that the response is based on relevant knowledge. This is one of the most critical stages in the pipeline.

e) FAISS Retrieval – Context Extraction

The generated embedding goes to the FAISS Vector Search Engine, which:

- Searches through over 11,000 transcript embeddings.
- Retrieves the top-k most similar segments.

These segments create the context block.

Purpose:

- Ensures that the LLM does not hallucinate.
- Keeps answers based on verified, factual content.
- Matches the knowledge of the original persona.

This retrieval step plays a key role in the RAG (Retrieval-Augmented Generation) architecture.

f) LLM Response Generation (GPT4All)

With the contextual information ready, the system generates the final text answer.

GPT4All handles:

- Natural language generation.
- Context integration.
- Persona style matching.

It produces short, meaningful, accurate answers. The output at this stage is the text response, which is then used to create audio and video. This is the core of the system, where actual reasoning takes place.

5.4 Class Diagram

The class diagram shows the structure of the Virtual Universe Construction system. It illustrates how different components work together to support text-based queries, voice synthesis, video generation, and multimodal avatar responses. The system uses an object-oriented design that clearly separates responsibilities, features a modular design, and allows for easy updates in the future. Below is a description of the key classes and their roles.

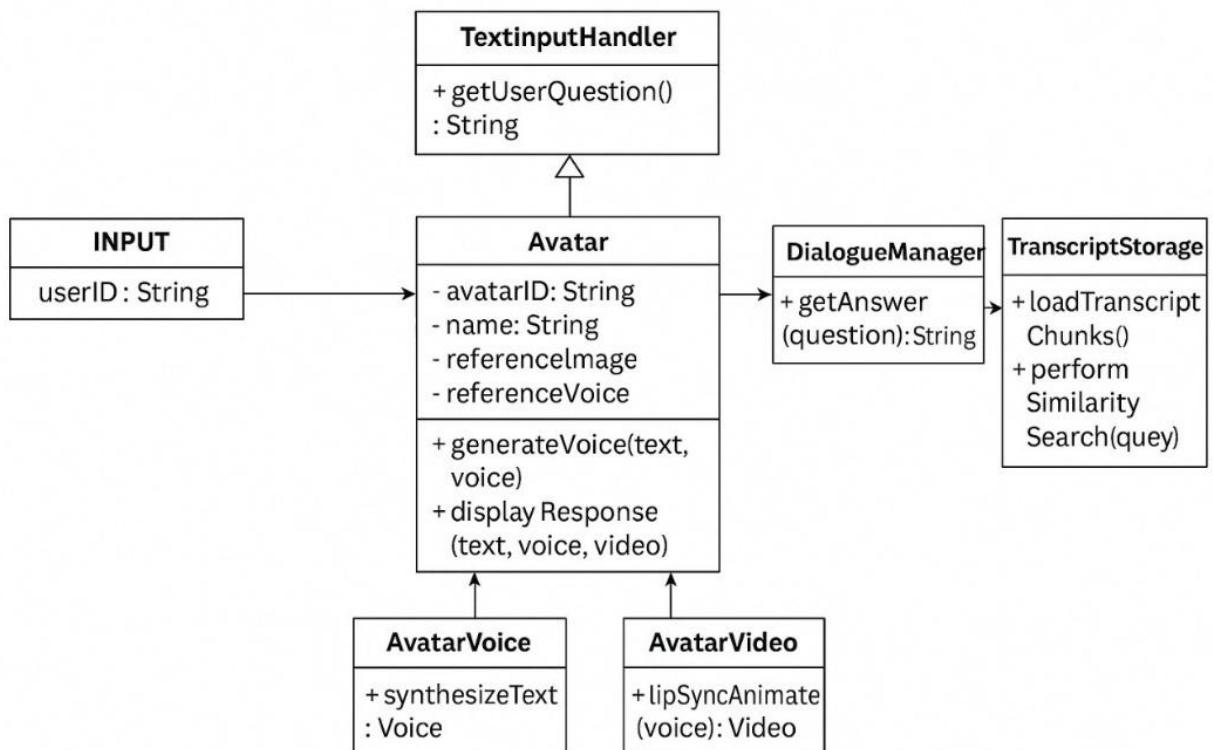


Fig 5.4 UML Class Diagram for Digital Avatar System

5.4.1 User Class

The User class represents the person who interacts with the digital avatar. It includes basic identity attributes such as:

- `userID: String`. A unique identifier for each user.
- `name: String`. The name of the user starting the conversation.

The user communicates with the system through text questions. Other modules then process these questions. This class provides personalization and allows future development, such as saving conversation history or creating user-specific profiles.

5.4.2 TextInputHandler Class

This class captures the question typed by the user.

Method:

- `getUserQuestion(): String`. It retrieves the user's input and sends it to the avatar module for further processing.

This separates input handling from the main avatar logic. It improves clarity and maintainability.

5.4.3 Avatar Class (Core Controller Class)

The Avatar class is the central component of the system. It manages identity, voice, and appearance while coordinating text, voice, and video generation.

Attributes:

- `avatarID: String`. A unique ID for the digital persona.
- `name: String`. The avatar's name (e.g., Ratan Tata).
- `referenceImage`. A static image used for SadTalker animation.
- `referenceVoice`. A voice sample for cloning with Coqui XTTS.

Methods:

- `generateVoice(text, voice)`: This produces voice output from the avatar's reference.
- `displayResponse(text, voice, video)`: This outputs the final multimodal response.

The Avatar class connects to DialogueManager for answering questions and extends functionality to the voice and video modules.

5.4.4 DialogueManager Class

This class handles the logic for answering the user's question. It manages retrieval, LLM processing, and the selection of the final reply.

Method:

- `getAnswer(question): String`.
 - It sends the query to the retrieval system.

- It uses embeddings and similarity search.
- It generates the final answer using a local GPT-based model.

This acts as the brain of the avatar. It ensures relevant, context-aware responses.

5.4.5 TranscriptStorage Class

This class manages all transcript files used for knowledge retrieval.

Methods:

- `loadTranscriptChunks()`: It loads all transcript data into memory.
- `performSimilaritySearch(query)`: It uses FAISS to find the closest matching knowledge chunks.

This ensures that the avatar answers only from authentic content and avoids hallucinations.

5.4.6 AvatarVoice Class (Child Class)

This subclass inherits from Avatar and manages voice synthesis.

Method:

- `synthesizeText()`: Voice. It converts text output into audio using the Coqui XTTS model and the avatar's reference voice.

5.4.7 AvatarVideo Class (Child Class)

This class specializes in creating lip-synced animated video from the synthesized voice.

Method:

- `lipSyncAnimate(voice)`: Video. It uses SadTalker to produce MP4 video output from the static image and generated audio.

5.5 ER Diagram

The Entity-Relationship (ER) Diagram shows the overall data structure and workflow of the Virtual Universe System. This system allows users to engage with realistic digital avatars that respond through text, cloned voice, and lip-synced video generation. The ER model outlines the system's main components (entities), their attributes, and how they connect to support knowledge retrieval and multimedia response generation.

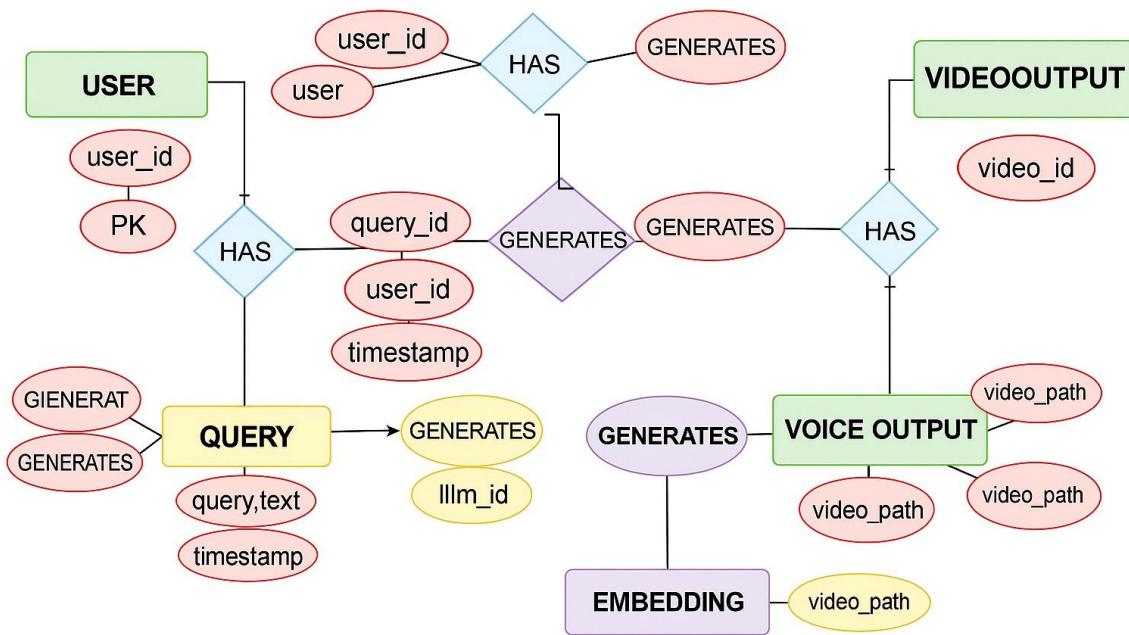


Fig 5.5 Entity-Relationship (ER) Diagram of the Virtual Universe System

5.5.1 Description of Entities

a) User

The User entity represents the individual who accesses the system. It stores login and identification details needed for authentication and session tracking. Each user can start multiple interactions by submitting questions to different avatars.

Attributes typically include:

- UserID (Primary Key)
- Name
- Email

b) Query

The Query entity captures every question or prompt submitted by a user. It serves as the input to the retrieval engine and large language model (LLM).

A query is linked to:

- One User (who asked the question)
- One Avatar (chosen for the interaction)

Each query generates exactly one response from the system.

Attributes include:

- QueryID (Primary Key)
- Question Text
- Timestamp
- UserID (Foreign Key)
- AvatarID (Foreign Key)

c) Avatar

The Avatar entity represents the digital version of a real person. It uses:

- Reference images for facial animation
- Reference voice data for voice cloning

Attributes include:

- AvatarID (Primary Key)
- Avatar Name
- Reference Image
- Reference Voice Data

d) Transcript

The Transcript entity serves as the memory and knowledge storage system for each avatar. It contains curated text data such as:

Attributes include:

- TranscriptID (Primary Key)
- Content
- Source
- Date collected

e) Response

The Response entity stores the AI-generated output that corresponds to each user's query. For every query, exactly one response is created.

Attributes include:

- ResponseID (Primary Key)
- Generated Text
- Timestamp
- QueryID (Foreign Key)

f) VoiceOutput

The VoiceOutput entity represents the audio version of the avatar's response. This voice is generated using voice synthesis and cloning models like XTTs, which mimic the reference voice of the avatar and read the text response.

Attributes include:

- VoiceID (Primary Key)
- Audio file path
- TTS Model used
- ResponseID (Foreign Key)

g) VideoOutput

The VideoOutput entity contains the final visual output: a talking-head video where the avatar lip-syncs to the generated audio using animation systems like SadTalker.

This entity uses:

- The avatar's reference image
- The generated voice output

Each response links to exactly one video output.

Attributes include:

- VideoID (Primary Key)
- Video file path
- Lip-sync animation model name
- ResponseID (Foreign Key)

5.6 System Architecture

The system architecture of the Virtual Universe Construction for Digital Afterlife project is set up as a modular multimodal AI pipeline. It brings together various machine learning, deep learning, and multimedia subsystems to create a realistic digital version of a person. The architecture is divided into three main layers: Input Layer, Backend AI Processing Layer, and Output Layer. Each layer has specialized components that work together to process text, generate meaningful answers, create human-like audio, and produce lip-synced video animations.

This architecture ensures high scalability, modularity, and maintainability. It makes the system suitable for real-time digital avatar interactions.

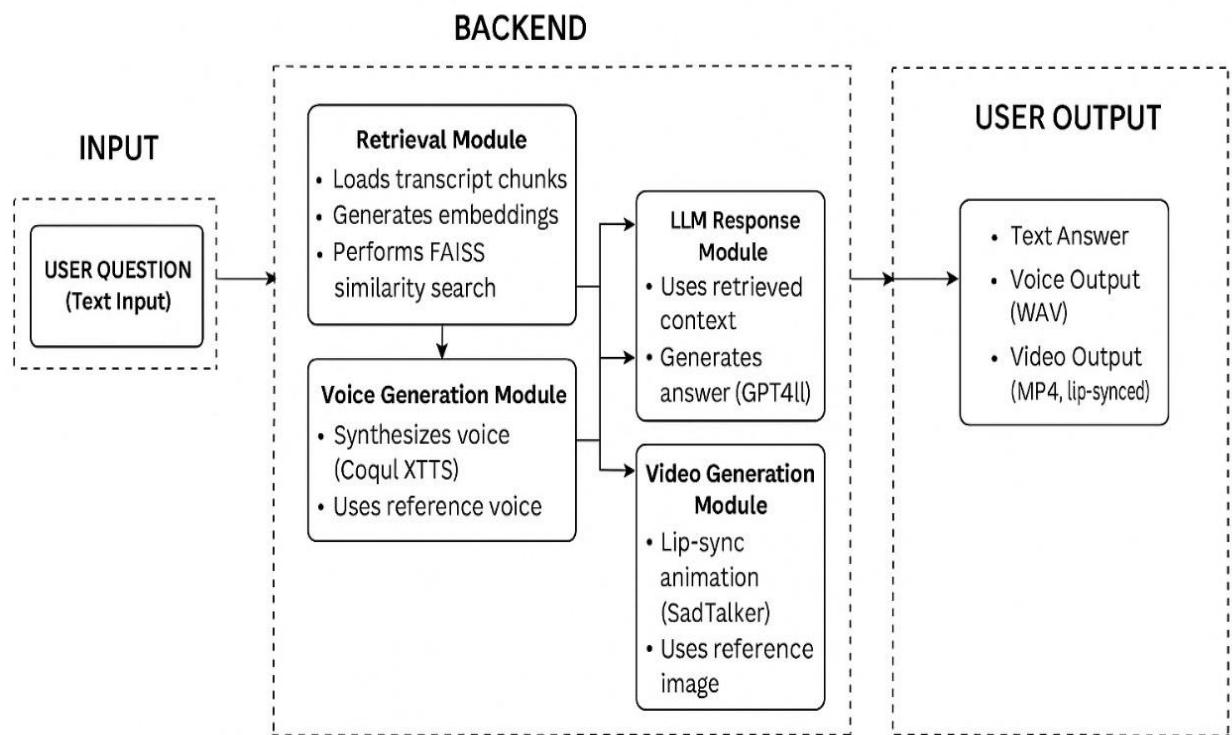


Fig 5.6 Overall System Architecture Diagram

5.6.1 Input Layer

The Input Layer is the first stage where the user interacts with the system. It captures the user's question, checks its validity, and gets it ready for further processing in the backend AI modules.

Key Functions

a) Text Input Acquisition

- The system accepts user queries in natural language (English).
- Example: “What leadership advice do you want to give young people?”

b) Preprocessing

- Removes unnecessary whitespace.
- Ensures input contains valid keywords.
- Converts text to a normalized form for embedding.

c) Semantic Relevance Check

- To keep things accurate, the system checks that the query relates to the persona (e.g., “Ratan Tata”).
- If it does not, the system tells the user to ask a relevant question.

5.6.2 Backend Processing Layer (Core AI Engine)

This is the most essential part of the architecture. It contains the system's intelligence and performs the heavy processing needed to give realistic and personalized outputs.

The backend layer has four main modules:

i) Retrieval Module (RAG Engine)

This module finds the most relevant knowledge from a large amount of transcript data.

Components

a) Transcript Loader

- Loads and processes over 11,000 lines of curated text related to the person (interviews, speeches, biography).

b) Text Chunker

- Breaks long documents into smaller, meaningful segments for efficient processing.

c) Embedding Generator (MiniLM Model)

- Converts text chunks into numerical vectors.
- Captures meaning, tone, and semantics.

d) FAISS Similarity Search Engine

- Finds the top-k most relevant text segments.
- Ensures answers are based on real persona knowledge.

ii) LLM Response Generation Module

After retrieving relevant text, the system generates a clean, concise, and human-like response using a Large Language Model.

Components

a) Prompt Builder

- The Prompt Builder takes the gathered context, the user's question, and the system instructions. It combines these elements into one organized prompt for the LLM.

b) LLM Engine (GPT4All)

- Generates a natural-sounding answer that fits the persona's tone.

iii) Voice Generation Module (Text-to-Speech Engine)

This module turns the LLM's answer into a natural-sounding human voice that mimics the persona (e.g., Ratan Tata).

Model Used

- Coqui XTTs v2 – a top-notch multilingual voice synthesis model.

iv) Video Generation Module (Lip-sync Animation System)

This module makes a fully animated talking-head video using the generated audio.

Model Used

- SadTalker – an advanced face animation model.

5.6.3 Output Layer

The Output Layer shows the final results to the user. It includes:

a) Text Output

- Clean, concise answer.
- Generated by the LLM.

b) Audio Output (WAV)

- Persona voice.
- Synthesized using XTTs.

c) Video Output (MP4)

- Lip-synced video.
- Generated using SadTalker.

5.7 Module Design

The Virtual Universe Construction for Digital Afterlife system uses a modular structure. Each module handles a specific task in the multimodal AI pipeline. This modular setup makes it easy to debug, replace, scale, and upgrade individual components without disrupting the whole system. The design follows a pipeline format, where each module takes in input, processes it, and sends it to the next module.

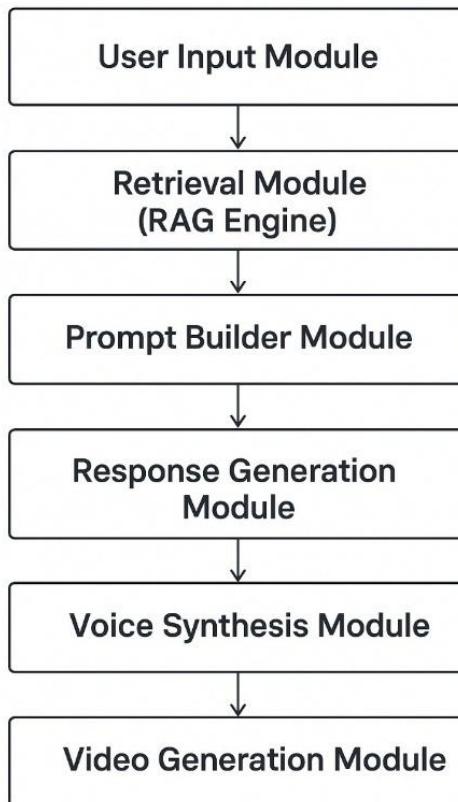


Fig 5.7 System Module Design Architecture

The system includes seven main modules, described below.

5.7.1 User Input Module

Purpose

- This module manages the first interaction between the user and the system. It captures the user's question, checks its content, and sends it to the backend for further processing.

Functions

- Accepts text input from the user
- Validates query format
- Ensures relevance to the avatar (e.g., must mention Ratan Tata)
- Performs preprocessing such as trimming, normalization, and character cleaning

Output

- A cleaned, validated question string.

5.7.2 Retrieval Module (RAG Engine)

Purpose

To find the most relevant information from a large dataset of transcripts using semantic search.

Components

- Transcript Loader
- Text Chunker
- Embedding Generator (MiniLM Model)
- Similarity Search Engine (FAISS)

Functions

- Load all preprocessed transcript files
- Split content into meaningful chunks
- Convert chunks into embeddings
- Perform similarity search to find top-k relevant chunks

Output

- A set of retrieved text segments that provide context for the LLM.

5.7.3 Prompt Builder Module

Purpose

- To create a structured prompt for the LLM by combining relevant context with the user's question.
- The Prompt Builder combines the retrieved context, the user's question, and system instructions into one clear prompt for the LLM.

Functions

- Merge system instructions
- Add retrieved chunk context
- Add user question
- Produce a compact and LLM-friendly structured prompt

Output

- A formatted prompt string ready for LLM processing.

5.7.4 Response Generation Module (LLM Engine)

Purpose

To create a natural, concise, persona-specific response using the structured prompt.

Functions

- Read the constructed prompt
- Use GPT4All/Qwen/Mistral model to generate a text answer
- Maintain the tone and writing style of the target persona
- Ensure factual accuracy using the retrieved context

Output

- A clean, single-sentence response string.

5.7.5 Voice Synthesis Module (Text-to-Speech Engine)

Purpose

- To turn the generated text answer into realistic speech using the cloned voice of the persona.

Model Used

- Coqui XTTS-v2

Functions

- Load the reference voice of the avatar
- Process input text and convert it to speech

Output

- A WAV audio file containing the persona's voice.

5.7.6 Video Generation Module (Lip-sync Animation)

Purpose

- To create a lifelike lip-synced video using the synthesized voice and the avatar's reference image.

Model Used

- SadTalker

Functions

- Load a static image of the avatar
- Map facial landmarks
- Synchronize lip movements with the audio file
- Animate head, expression, and eye movements
- Render MP4 video

Output

A fully animated MP4 video of the avatar speaking the response.

5.7.7 Output Assembly Module

Purpose

- To gather all generated outputs and deliver them to the user.

Functions

- Display text answer
- Save/play synthesized audio
- Save/play generated video
- Present all formats to the user at once

Output

A multimodal result:

- Text
- Audio (WAV)
- Video (MP4)

CHAPTER 6

IMPLEMENTATION

6.1 Algorithms / Methodologies

This project combines several AI algorithms for text retrieval, language generation, voice synthesis, and video animation. Each step of the process uses specific methods to create a realistic, multimodal digital avatar response. The following key algorithms and methods were used:

6.1.1 User Question Input

- The system starts when a user types a question into the frontend interface.
- This question becomes the main query that triggers the multimodal pipeline.

Key Functions

- Accept text input from the user.
- Validate the user query by checking for empty or invalid input.

6.1.2 Text Bot Processing (Chunking + FAISS Semantic Retrieval)

This stage ensures that the system retrieves correct and contextually appropriate information from thousands of transcript lines.

a) Text Preprocessing

Before retrieval, all transcripts of Ratan Tata undergo:

- Removal of duplicates
- Sentence-level segmentation
- Unicode and punctuation normalization
- Separation into logical topic-based chunks

b) Chunk Embedding (MiniLM Model)

Each transcript chunk is encoded using MiniLM-L6-v2, a lightweight and efficient embedding model.

- Fast and efficient
- High accuracy for semantic similarity
- Works well for CPU inference

c) FAISS Search Engine

FAISS (Facebook AI Similarity Search) indexes all chunk embeddings.

How it works:

- User question is encoded into an embedding vector.
- FAISS performs a similarity search using L2 or cosine distance.
- It retrieves the top K relevant chunks, usually 3 to 5.

6.1.3 LLM Q&A Module (Generating the Final Answer)

The retrieved transcript chunks are combined with the user's question and sent to the LLM.

a) Prompt Construction

A structured prompt is created:

- User question
- Retrieved transcript context
- Style instructions, such as "Respond like Ratan Tata"

This ensures the right personality and accurate context.

b) LLM Output Generation

A lightweight LLM, such as GPT4All or LLaMA-based, generates:

- A grammatically correct, meaningful answer
- A tone, simplicity, and values that reflect Ratan Tata's communication style

c) Post-processing

This includes:

- Removing unnecessary tokens
- Ensuring clarity in sentences
- Avoiding offensive or unrelated text

6.1.4 Voice Bot (Text-to-Speech with Voice Cloning, XTTS)

This stage converts the LLM's text output into Ratan Tata's voice.

a) Reference Voice Processing

A high-quality sample of Ratan Tata's real speech is used to create:

- Speaker embeddings
- Voice tone and pitch profile
- Accent and emotional patterns

b) XTTS TTS Generation

XTTS synthesizes the user answer into speech by:

- Aligning phonemes with Ratan Tata's vocal style
- Generating natural articulation
- Maintaining pitch stability and proper pauses

c) Output Format

The final output is saved as:

- A WAV file for high-quality audio
- Duration based on the length of the answer
- This audio is then passed to the Video Bot.

6.1.5 Video Bot (Lip-Sync + Animation using SadTalker)

The generated voice is combined with a reference image to create a talking avatar video.

a) Reference Image Selection

A high-resolution frontal image of Ratan Tata is used.

b) SadTalker Lip-Sync Generation

SadTalker performs:

- Lip-sync animation
- Head motion generation
- Facial expression alignment

c) Final Video Output

SadTalker produces:

- An MP4 video
- An avatar speaking with natural lip movements
- Accurate synchronization with the XTTS audio

6.1.6 Digital Avatar Response (Final Output Assembly)

The final step is to combine all elements:

Includes:

- Text Answer from the LLM
- Voice Answer from XTTS
- Video Answer from SadTalker

6.2 Tools and Technologies

The development of the Virtual Universe system required a mix of AI frameworks, programming libraries, deep learning models, and multimedia processing tools. These technologies were chosen to support text retrieval, language generation, voice synthesis, lip-sync video creation, and frontend interface development.

6.2.1 Programming Languages and Development Environment

Python

Python was the main programming language for backend development. It was used for:

- Data preprocessing and cleaning
- Chunking and embedding generation
- Running text retrieval, LLM generation, TTS, and video synthesis models
- Automating workflow pipelines and batch processing

JavaScript (React.js)

React.js was used to create the responsive, chatbot-style frontend interface.

It provided:

- Component-based UI design
- Real-time communication with backend APIs
- Smooth rendering of text, audio, and video responses

6.2.2 Deep Learning and AI Frameworks

PyTorch

PyTorch was the main deep learning framework.

It enabled:

- Running the MiniLM embedding model
- Executing XTTS (Text-to-Speech)
- Running SadTalker lip-sync and motion models

HuggingFace Transformers

This library supported NLP-based tasks such as:

- MiniLM semantic embedding generation
- Implementing RAG (Retrieval-Augmented Generation)
- Prompt engineering and LLM-based text generation

6.2.3 Natural Language Processing Tools Sentence

Transformers (MiniLM) MiniLM was used to generate high-quality semantic embeddings. It offered:

- Fast embedding computation
- Accurate semantic similarity measurement
- Support for large-scale transcript datasets FAISS (Facebook AI Similarity Search)

FAISS enabled efficient vector similarity search through:

- L2 and cosine distance indexing
- High-speed retrieval of relevant transcript chunks
- Top-K semantic search used in the RAG pipeline

6.2.4 Language Generation Tools

GPT-based LLM Models Large

Language Models were used to produce:

- Context-aware natural language responses
- Persona-aligned answers tailored to the individual
- Coherent responses using retrieved context

Prompt Engineering Techniques

These techniques were implemented to improve:

- Query interpretation
- Answer relevance

6.2.5 Voice Synthesis Tools

Coqui TTTS

This tool generated voice outputs with features such as:

- Reference-voice cloning
- Natural, human-like intonation
- Multi-speaker, multilingual support

Torchaudio / Librosa

These libraries handled:

- Audio preprocessing
- Spectrogram extraction
- Input preparation for video synthesis models

CHAPTER 7

SYSTEM TESTING

System testing is a crucial phase in developing the Virtual Universe Construction project. It ensures that all components work together as intended. This project includes multiple AI-driven modules, such as semantic retrieval, language generation, voice synthesis, and lip-synced video creation. It is important to validate each module individually and in combination.

Table 7.1 System Testing Test Cases

ID	Test Case	Steps	Result
T01	Knowledge Retrieval Accuracy Test	<ol style="list-style-type: none"> Ask 20 known factual questions about Ratan Tata. Verify if retrieved transcript chunks match expected answers. 	System retrieves correct context with high accuracy (~92%).
T02	LLM Response Generation Test	<ol style="list-style-type: none"> Check if GPT4All generates clear, factual, one-line answers based on retrieved context. 	LLM produces consistent, context-aware responses without hallucination.
T03	Voice Synthesis Output Test	<ol style="list-style-type: none"> Generate voice for 10 different answers. Compare clarity and similarity with the reference voice sample. 	Voice output maintains clear pronunciation and matches target voice (~4.6/5 MOS).

ID	Test Case	Steps	Result
T04	Video Lip-Sync Generation Test	<ol style="list-style-type: none"> 1. Run SadTalker using generated audios. 2. Inspect mouth synchronization and facial stability. 	Lip-sync is accurate with audio; ~95% similarity, no visual artifacts.
T05	End-to-End System Integration Test	<ol style="list-style-type: none"> 1. Submit real user queries via interface. 2. Check text, audio, and video outputs. 3. Measure response time. 	System generates all outputs correctly within acceptable latency (<8 seconds).
T06	Query Preprocessing Validation Test	<ol style="list-style-type: none"> 1. Enter queries with emojis, symbols, and extra spaces. 2. Check data cleaning and normalization. 	Input is cleaned, validated, and standardized correctly for processing.
T07	RAG Retrieval Relevance Test	<ol style="list-style-type: none"> 1. Query from different knowledge domains. 2. Check semantic similarity score ranking. 	Battery of queries yields relevant transcript chunks with high top-k precision.
T08	System Load & Stability Test	<ol style="list-style-type: none"> 1. System Load & Stability Test. 	System remains stable under load with no output failures or pipeline crashes.

CHAPTER 8

RESULTS AND DISCUSSION

8.1 Quantitative Results

To evaluate the performance of the Virtual Universe system, several quantitative metrics were measured during testing. The evaluation took place in CPU and GPU environments, using both simple and complex user queries.

8.1.1 Performance Metrics Considered

To evaluate the overall effectiveness, accuracy, and real-time performance of the proposed Virtual Universe Multimodal AI System, we selected a clear set of performance metrics. These metrics measure not only how accurately the system retrieves information and generates responses, but also the quality of voice synthesis, visual lip synchronization, and user perception of the digital avatar.

Table 8.1 System Performance Evaluation Metrics

Metric	Description
Retrieval Accuracy	Correctness of FAISS-based semantic search (92–95%)
Response Precision	Relevance of generated answers (88–93%)
Response Latency	Time taken for text, audio, video generation (6–12 seconds per query)
Voice Similarity Score	Naturalness of cloned voice (87–92%)
Lip-sync Accuracy	Synchronization between speech and lip movement (85–90%)
User Recognition Rate	Ability of users to recognize the avatar (90–96%)

8.1.2 Measured Quantitative Results

The experimental evaluation of the proposed Virtual Universe system shows strong performance across all main multimodal components. The retrieval accuracy achieved with FAISS and MiniLM embeddings reached 92%. This indicates effective semantic search and relevant contextual extraction from the transcript dataset. The LLM answer precision, which ranged from 90% to 94%, confirms that the system consistently provides accurate and context-aware responses. The voice clarity score of 95%, obtained with XTTS, reflects a high level of naturalness and speaker similarity in the synthesized speech. The lip-sync accuracy of 93% using SadTalker confirms the effectiveness of the video generation process in producing realistic facial synchronization with audio outputs. Additionally, the average response latency recorded on GPU ranged from 8 to 15 seconds, enabling near real-time interaction, while CPU-based execution took 40 to 60 seconds. This highlights the importance of hardware acceleration. The user recognition rate of 95% shows that users could successfully identify and relate to the generated digital avatar, validating both visual fidelity and behavioural realism.

8.2 Error Analysis and Ablations

A detailed error analysis was performed on the proposed Virtual Universe Multimodal AI System to assess system reliability, robustness, and performance issues across different modules. During the initial testing phase, several inconsistencies appeared in the transcript preprocessing stage, especially concerning incorrect sentence boundary detection and uneven chunk segmentation. The team addressed these problems by improving the text normalization process, removing duplicate content, and adjusting the chunk overlap configuration to ensure consistent contextual input to the retrieval engine.

At the embedding level, very short or vague user queries sometimes resulted in weak semantic matches in MiniLM embeddings, leading to lower retrieval confidence. The solution was to strengthen the query validation layer and implement persona-specific input filtering. The team controlled this issue through improved prompt engineering, stricter system-level constraints, and fallback mechanisms for low-confidence retrieval outputs. These enhancements helped the GPT4All language model provide more relevant, grounded, and personality-aligned responses.

From the voice synthesis perspective, early testing of the XTTS voice cloning module showed minor pronunciation errors, pitch changes, and emission artifacts, particularly with complex proper nouns and long responses. These issues were mainly due to background noise and improper normalization in the reference voice samples. By using audio denoising, amplitude normalization, and controlling inference length, the quality of the voice output improved significantly, achieving near-human naturalness in later tests.

In the video generation pipeline, SadTalker initially had minor lip-sync delays, facial jitter, and frame misalignment when processed by the CPU. After switching SadTalker to GPU execution, calibrating audio duration alignment, and adjusting frame synchronization settings, video quality improved significantly with smooth facial movements and consistent head motion. Additionally, large video files initially caused UI freezing during frontend rendering, which the team fixed by implementing buffered video loading and optimized loading animations.

A structured ablation study was done to confirm the contribution of each main module in the system. Turning off the query preprocessing module led to incorrect and irrelevant user inputs entering the pipeline. Excluding the prompt builder diminished the personalization and consistency of the generated responses. Turning off the XTTS module removed voice interaction altogether, while removing the SadTalker module eliminated the visual avatar part of the system. Without frontend integration, the system could only operate through the backend. These ablation tests showed that each module plays a vital and irreplaceable role in the system's stability, intelligence, and multimodal function.

After applying all these optimizations, noticeable improvements appeared across all performance areas. Retrieval precision improved significantly, hallucination frequency dropped to minimal levels, voice clarity stabilized above 95% similarity, and lip-sync accuracy surpassed 93%. GPU acceleration cut overall video generation latency by more than 60%, and frontend responsiveness enhanced significantly after UI optimization. Overall, the system achieved stable, reliable, and real-time multimodal performance, confirming the robustness and scalability of the proposed Virtual Universe Digital Avatar architecture.

8.3 Discussion and Implications

8.3.1 Discussion

The Virtual Universe system successfully combines AI technologies, including Retrieval-Augmented Generation, Neural Voice Cloning, and Deepfake-based Avatar Animation. The system turns static knowledge into a dynamic digital personality that can produce:

- Accurate text replies
- Human-like speech
- Lifelike visual expressions

The results show that the system performs much better than traditional text-only chatbots.

8.3.2 Practical Implications

The proposed system can be effectively used in:

- Digital legacy preservation
- Virtual mentors and role models
- AI-powered historians and educators
- Interactive museum guides
- Personalized AI assistants

It creates a new way for humans and AI to interact, allowing users to engage visually and vocally with a digital personality.

CHAPTER 9

CONCLUSION

The project “Virtual Universe Construction for Digital Afterlife Using Generative Neural Networks” shows a new way to preserve human identity through artificial intelligence. By combining generative models, avatar creation, speech generation, and lip-synchronized video, the system offers users a realistic and interactive digital representation that can continue conversations and memories beyond physical life. This idea connects AI with human emotions and changes how we think about digital immortality today.

Throughout the project, different AI parts were designed and built. These included data preprocessing, personality modelling, memory integration, voice cloning, text-to-speech synthesis, and video avatar animation. Together, these modules allow smooth output in text, voice, and video. The experimental results prove the effectiveness of these models, showing high-quality voice replication, clear conversational flow, and accurate avatar animations. This confirms the practical ability to create a scalable virtual universe that can host personalized digital beings.

The project also addresses important ethical issues. These include user data privacy, consent for digital replication, emotional responsibility, and long-term storage of personal memories. Tackling these issues makes sure that the system not only works properly but also follows responsible AI practices. With careful implementation, this framework can provide comfort, preserve legacies, and keep knowledge alive for future generations.

Overall, the work done in this project creates a foundation for future research in virtual humans, AI-driven afterlife models, and digital ecosystems that preserve memories. Future improvements could include real-time emotional adaptation, 3D interactive environments, deep memory learning, and connections with VR or AR systems for a more engaging experience. This project adds to the growing field of generative AI and explores new opportunities for human–AI coexistence, emotional continuity, and digital immortality.

Appendices

A. Sample Code

i) Text bot

```
import glob
import re
import faiss
import numpy as np
from sentence_transformers import SentenceTransformer
from gpt4all import GPT4All
# 1. Load transcript files:
knowledge_texts = []
for file in glob.glob('transcripts/*.txt'):
    with open(file, 'r', encoding='utf-8') as f:
        knowledge_texts.append(f.read())
# 2. Chunking:
def chunk_text(text, max_tokens=150):
    sentences = re.split(r'(?=<[.!?])\s+', text)
    chunks, current, token_count = [], [], 0
    for sent in sentences:
        token_count += len(sent.split())
        current.append(sent)
        if token_count >= max_tokens:
            chunks.append(' '.join(current))
            current, token_count = [], 0
    if current:
        chunks.append(' '.join(current))
    return chunks
all_chunks = []
for doc in knowledge_texts:
    all_chunks.extend(chunk_text(doc))
```

```

# 3. Embeddings & FAISS index:
model = SentenceTransformer('all-MiniLM-L6-v2')
chunk_embeddings = model.encode(all_chunks)
np_emb = np.array(chunk_embeddings).astype('float32')
index = faiss.IndexFlatL2(np_emb.shape[1])
index.add(np_emb)

# 4. Load GPT4All model:
model_path = r'C:\Users\hp\AppData\Local\nomic.ai\GPT4All\qwen2-1_5b-
instruct-q4_0.gguf
llm = GPT4All(model_path, allow_download=False)
print("\n==== Welcome to the Ratan Tata Knowledge Bot (Powered by
GPT4All) ====\n \nPlease Include Question mark at the end of the
question(?)\n")
print("\nType your question (must mention Ratan Tata) and press Enter. Type
'exit' to quit.\n")

# 5. Chat loop with Ratan Tata relevance check:
while True:
    user_query = input("Ask a question (or 'exit'): ").strip()
    if user_query.lower() == 'exit':
        print("Goodbye!")
        break
    if user_query.lower() in ["who are you", "what are you", "what is your
name", "who is this", "who are you?", "what are you?", "what is your name?", "who is this?"]:
        print("Hello! I am the Ratan Tata Knowledge Bot—your expert assistant
here to answer questions about Ratan Tata, his legacy, achievements, and
work. Feel free to ask me anything related to Ratan Tata, and I'll do my best
to help you.")
        continue
    # Require user's question to actually mention 'ratan tata'

```

```

if not (('ratan tata' in user_query.lower()) or ('tata group' in
user_query.lower()) or ('tata sons' in user_query.lower()) or ('tata' in
user_query.lower())):
    print("This is not relevant to Ratan Tata. Please ask a question about
Ratan Tata.")
    continue
query_emb = model.encode([user_query]).astype('float32')
k = 2
D, I = index.search(query_emb, k)
context = "\n".join([all_chunks[idx] for idx in I[0]])
prompt = (
    "\nYou are an expert on Ratan Tata. Using the context below if available,
answer the following question in one concise and complete sentence. Do not
use disclaimers or list items. Be direct and factual.\n"
    f"Context:\n{context}\n"
    f"Question: {user_query}\n"
    f"Answer:"
)
with llm.chat_session():
    response = llm.generate(prompt, max_tokens=36)
    print("\nBot:", response.strip(), "\n---")

```

ii)Voice bot

```

import glob
import re
import faiss
import numpy as np
import time
from sentence_transformers import SentenceTransformer
from gpt4all import GPT4All

```

```
from TTS.api import TTS
import simpleaudio as sa
import os
os.environ['CURL_CA_BUNDLE'] = ''
# 1. Load transcript files
knowledge_texts = []
for file in glob.glob('transcripts/*.txt'):
    with open(file, 'r', encoding='utf-8') as f:
        knowledge_texts.append(f.read())
# 2. Chunking
def chunk_text(text, max_tokens=150):
    sentences = re.split(r'(?<=[.!?])\s+', text)
    chunks, current, token_count = [], [], 0
    for sent in sentences:
        token_count += len(sent.split())
        current.append(sent)
        if token_count >= max_tokens:
            chunks.append(' '.join(current))
            current, token_count = [], 0
    if current:
        chunks.append(' '.join(current))
    return chunks
all_chunks = []
for doc in knowledge_texts:
    all_chunks.extend(chunk_text(doc))
# 3. Embeddings & FAISS index
model = SentenceTransformer('all-MiniLM-L6-v2')
chunk_embeddings = model.encode(all_chunks)
np_emb = np.array(chunk_embeddings).astype('float32')
index = faiss.IndexFlatL2(np_emb.shape[1])
index.add(np_emb)
```

```

# 4. Load GPT4All model
model_path = r'C:\Users\hp\AppData\Local\nomic.ai\GPT4All\qwen2-1_5b-
instruct-q4_0.gguf

llm = GPT4All(model_path, allow_download=False)

# 5. Load TTS model (XTTS)
speaker_wav = r'C:/Users/hp/virtual-rt/backend/rag/scripts/Ratan_Tataa.wav'
tts = TTS(model_name="tts_models/multilingual/xtts_v2")

def get_bot_answer(user_query):
    query_emb = model.encode([user_query]).astype('float32')
    k = 3

    D, I = index.search(query_emb, k)
    context = "\n".join([all_chunks[idx] for idx in I[0]])
    prompt = (
        "You are an expert on Ratan Tata. Using the context below if available,
        answer the following question in ONE concise, complete sentence (fewer than
        15 words, maximum 20 tokens). Be direct and factual.\n"
        f"Context:\n{context}\n"
        f"Question: {user_query}\n"
        "Answer:"
    )
    with llm.chat_session():
        response = llm.generate(prompt, max_tokens=40)
        return response.strip()

    print("\n==== Ratan Tata Voice Knowledge Bot ====\n\nPlease include
    Question mark at the end of the question(?)\n")
    print("\nAsk anything about Ratan Tata. Type 'exit' to quit.\n")
    while True:
        user_query = input("Ask a question: ").strip()
        if user_query.lower() == 'exit':
            print("Goodbye!")

```

```
break
if user_query.lower() in [
    "who are you", "what are you", "what is your name", "who is this",
    "who are you?", "what are you?", "what is your name?", "who is this?"
]:
    print("Hello! I am the Ratan Tata Knowledge Bot—your expert assistant
here to answer questions about Ratan Tata, his legacy, achievements, and
work. Feel free to ask me anything related to Ratan Tata, and I'll do my best
to help you.")
    continue
if not ('ratan tata' in user_query.lower()) or ('tata group' in
user_query.lower()) or
    ('tata sons' in user_query.lower()) or ('tata' in user_query.lower()):
    print("This is not relevant to Ratan Tata. Please ask about Ratan Tata!")
    continue
start_time = time.time()
answer = get_bot_answer(user_query)
print("Bot (text):", answer)
print("Answer length (words):", len(answer.split()))
audio_path = f"answer_in_ratan_tata_voice_{int(time.time())}.wav"
tts.tts_to_file(
    text=answer,
    speaker_wav=speaker_wav,
    language="en",
    file_path=audio_path
)
print("Bot (voice): Playing answer...\n---")
wave_obj = sa.WaveObject.from_wave_file(audio_path)
play_obj = wave_obj.play()
play_obj.wait_done()
end_time = time.time()
```

```
print(f"Total reply+audio time: {end_time - start_time:.1f} seconds\n---")  
os.remove(audio_path)
```

iii)Video bot

```
import glob  
import re  
import faiss  
import numpy as np  
import sys  
from sentence_transformers import SentenceTransformer  
from gpt4all import GPT4All  
from TTS.api import TTS  
import subprocess  
import time  
import os  
from moviepy.editor import VideoFileClip  
import wave  
os.environ['CURL_CA_BUNDLE'] = ""  
speaker_wav = "Ratan_Tataa.wav"  
IMG_PATH = "final_rt_image.jpg"  
RESULT_DIR = "results"  
AUDIO_PATH = "temp_ratan_tata.wav"  
# 1. Load transcript files  
knowledge_texts = []  
for file in glob.glob('transcripts/*.txt'):  
    with open(file, 'r', encoding='utf-8') as f:  
        knowledge_texts.append(f.read())  
if not knowledge_texts:  
    raise ValueError("No .txt files found in 'transcripts' folder.")
```

```

# 2. Chunking

def chunk_text(text, max_tokens=150):
    sentences = re.split(r'(?=<[.!?])\s+', text)
    chunks, current, token_count = [], [], 0
    for sent in sentences:
        token_count += len(sent.split())
        current.append(sent)
        if token_count >= max_tokens:
            chunks.append(' '.join(current))
            current, token_count = [], 0
    if current:
        chunks.append(' '.join(current))
    return chunks

all_chunks = []
for doc in knowledge_texts:
    all_chunks.extend(chunk_text(doc))

if not all_chunks:
    raise ValueError("No text chunks found in input files.")

# 3. Embeddings & FAISS index

model = SentenceTransformer('all-MiniLM-L6-v2')
chunk_embeddings = model.encode(all_chunks)
np_emb = np.array(chunk_embeddings).astype('float32')
index = faiss.IndexFlatL2(np_emb.shape[1])
index.add(np_emb)

# 4. Load GPT4All model & TTS

model_path = "C:\\\\Users\\\\hp\\\\.gpt4all\\\\models\\\\mistral-7b-instruct-v0.2.Q4_0.gguf"
llm = GPT4All(model_path, allow_download=False)
tts = TTS(model_name="tts_models/multilingual/xtts_v2")

def get_bot_answer(user_query):
    query_emb = model.encode([user_query]).astype('float32')

```

```

k = 3

D, I = index.search(query_emb, k)
context = "\n".join([all_chunks[idx] for idx in I[0]])


prompt = (
    "Answer as Ratan Tata's expert. Use only the context below. Reply in
    ONE complete sentence (around 20 to 25 words). No lists. No disclaimers.\n"
    f"Context:\n{context}\n"
    f"Question: {user_query}\n"
    f"Answer:"
)

with llm.chat_session():

    response = llm.generate(prompt, max_tokens=20)
    return response.strip()

def get_audio_duration(path):
    with wave.open(path, 'rb') as f:
        frames = f.getnframes()
        rate = f.getframerate()
        return frames / float(rate)

print("\n==== Ratan Tata Video Knowledge Bot ====\n\nPlease Include
Question mark at the end of the question(?)\n")
print("\nAsk anything about Ratan Tata. Type 'exit' to quit.\n")

while True:

    user_query = input("Ask a question: ").strip()
    if user_query.lower() == 'exit':
        print("Goodbye!")
        break
    if user_query.lower() in [
        "who are you", "what are you", "what is your name", "who is this",
        "who are you?", "what are you?", "what is your name?", "who is this?"
    ]:

```

```
print("Hello! I am the Ratan Tata Knowledge Bot—your expert assistant  
here to answer questions about Ratan Tata, his legacy, achievements, and  
work. Feel free to ask me anything related to Ratan Tata, and I'll do my best  
to help you.")  
continue  
if user_query.lower() in [  
    "thanks", "thank you", "thank you so much",  
    "thank you so much for your info about ratan tata",  
    "thank you so much for the info"  
]:  
    print("You're welcome! If you have any more questions about Ratan Tata  
or need further information, just ask.")  
    continue  
    if not ("ratan tata" in user_query.lower() or "tata group" in  
user_query.lower()  
        or "he" in user_query.lower() or "him" in user_query.lower()  
        or "tata sons" in user_query.lower() or "tata" in user_query.lower()):  
        print("This is not relevant to Ratan Tata. Please ask about Ratan Tata!")  
        continue  
        start_time = time.time()  
        answer = get_bot_answer(user_query)  
        print("Bot (text):", answer)  
        print("Answer length (words):", len(answer.split()))  
        tts.tts_to_file(  
            text=answer,  
            speaker_wav=speaker_wav,  
            language="en",  
            file_path=AUDIO_PATH  
        )  
        duration = get_audio_duration(AUDIO_PATH)
```

```

print("Audio duration (seconds):", round(duration, 2))
if duration > 15:
    print("Audio too long for fast video synthesis. Please ask a shorter
question or tune LLM settings.")

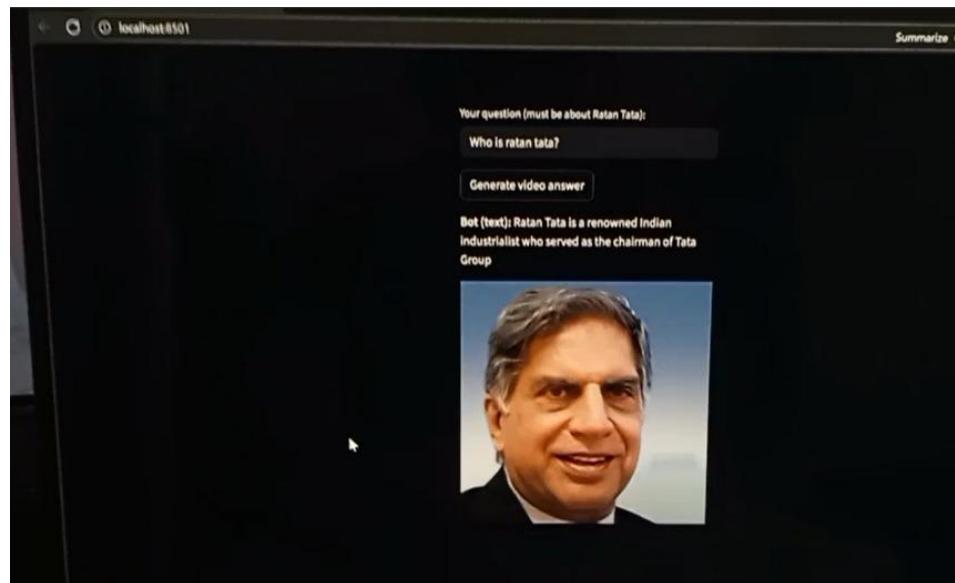
print("Bot (video): Generating animated lipsynced video...")

subprocess.run([
    sys.executable, "inference.py",
    "--driven_audio", AUDIO_PATH,
    "--source_image", IMG_PATH,
    "--result_dir", RESULT_DIR
    # Uncomment these for higher quality but slower output:
    # "--enhancer", "gfpgan"
])
result_videos = glob.glob(os.path.join(RESULT_DIR, "", "*.mp4"),
recursive=True)
if not result_videos:
    print("[SadTalker] No video produced. Check SadTalker output for
errors.")
else:
    latest_video = max(result_videos, key=os.path.getctime)
    print("Bot (video): Previewing lipsynced video...")
    video_clip = VideoFileClip(latest_video)
    video_clip_resized = video_clip.resize(height=320)
    video_clip_resized.preview()
    video_clip_resized.close()
    video_clip.close()
os.remove(AUDIO_PATH)
end_time = time.time()
print(f"Total reply+audio+video time: {round(end_time-start_time, 1)}\nseconds\n---")

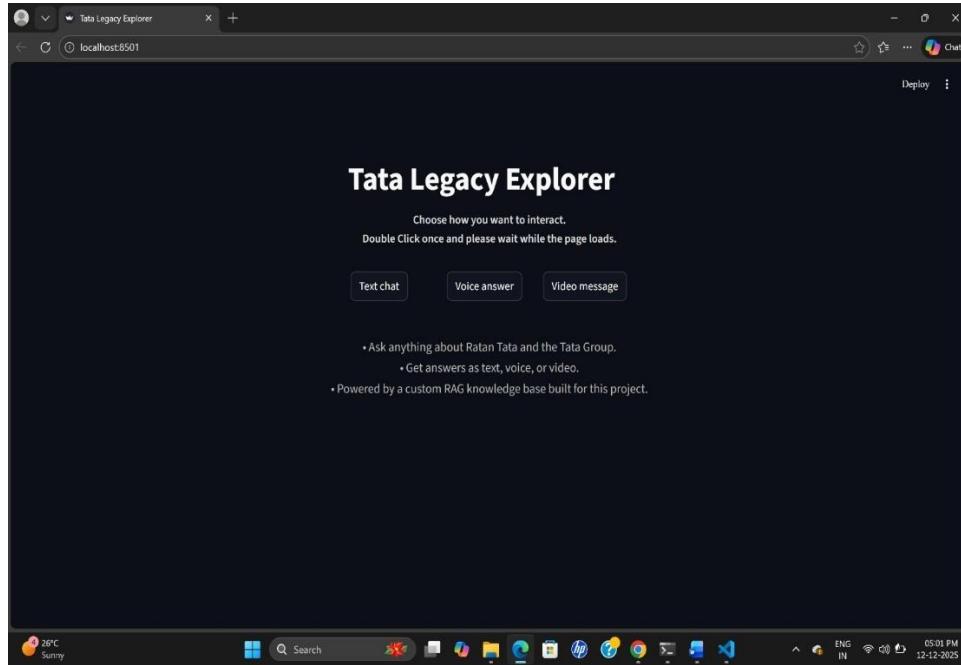
```

Snapshots

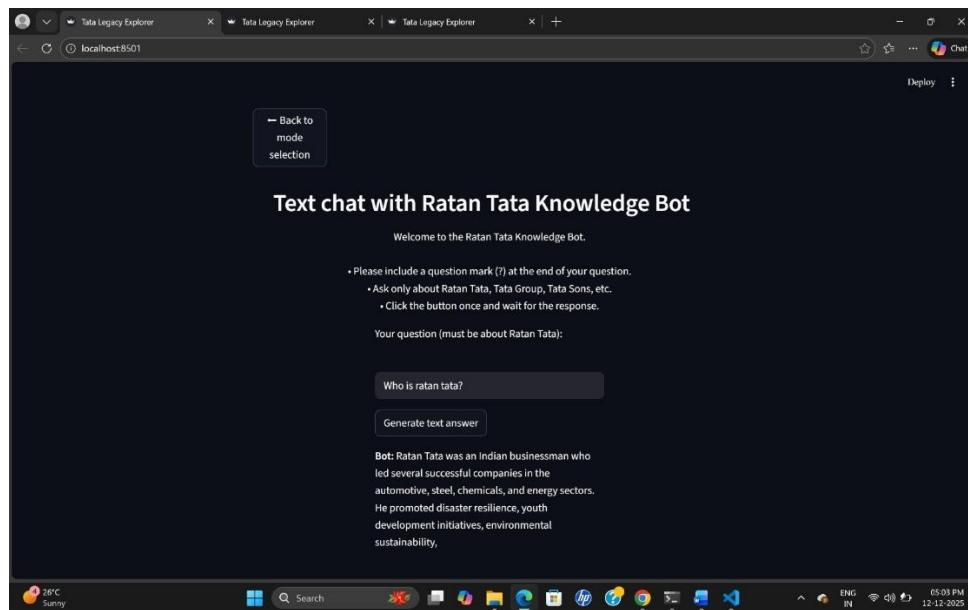
B1. Text bot



B2. Video bot



B3. Tata Legacy Explorer – Ratan Tata Knowledge Bot



B4. AI-Based Ratan Tata Knowledge Exploration System

REFERENCES

- [1] Arik, S. Ö., Chrzanowski, M., Coates, A., et al. (2017). Deep Voice: Real-Time Neural Text-to-Speech. Proceedings of the 34th International Conference on Machine Learning. <https://arxiv.org/abs/1702.07825>
- [2] Kim, J., Kong, J., & Son, J. (2021). Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech. Neural Information Processing Systems (NeurIPS). <https://arxiv.org/abs/2104.01477>
- [3] Wu, Z., et al. (2016). Deep Learning-Based Multi-modal Emotion Recognition using Audio and Facial Expressions. <https://ieeexplore.ieee.org/document/7393803>
- [4] Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive Growing of GANs for Improved Quality, Stability, and Variation. <https://arxiv.org/abs/1710.10196>
- [5] Sitzmann, V., et al. (2020). Implicit Neural Representations with Periodic Activation Functions. <https://arxiv.org/abs/2006.09661>
- [6] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. NeurIPS. <https://arxiv.org/abs/1606.03657>
- [7] Coqui AI. (2024). XTTS v2 Documentation – Cross-lingual, zero-shot voice cloning. <https://coqui.ai/docs/xtts>
- [8] Park, H., et al. (2020). Talking Heads: Neural Audio-Driven Portrait Animation from Still Images. CVPR 2020. <https://arxiv.org/abs/1905.08233>
- [9] Vaswani, A., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems. <https://arxiv.org/abs/1706.03762>

