*(Page Break)*

AI Mirror: A Multi-Modal System for Personalized AI Persona Simulation

**By:**

Anas Saleh

Mennatullah Essam

Yara Mohammed

Mahmoud Nasr

Mohamed Ashraf

Mohamed Ehab

Supervised By:

Dr. Salwa Osama

*A project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science*

**(Insert University Logo Here)**

Faculty of Computers and Artificial Intelligence

[Your Department Name]

[Your University Name]

June 2025

*(Page Break)*

Approval Page

**Project Title:** AI Mirror: A Multi-Modal System for Personalized AI Persona Simulation

The undersigned have read and approved this project submitted by the students listed below as a partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

**Team Members:**

* Anas Saleh
* Mennatullah Essam
* Yara Mohammed
* Mahmoud Nasr
* Mohamed Ashraf
* Mohamed Ehab

**Signatures:**

Dr. Salwa Osama

(Project Supervisor)

[Name of Examiner 1]

(Examiner)

[Name of Examiner 2]

(Examiner)

[Name of Department Head]

(Head of Department)

*(Page Break)*

Dedication

To our families, for their endless support, encouragement, and patience throughout this challenging and rewarding journey. This work is a testament to your unwavering belief in us.

*(Page Break)*

Acknowledgement

We would like to express our deepest gratitude to our project supervisor, Dr. Salwa Osama, for her invaluable guidance, expertise, and insightful feedback throughout the development of this project. Her direction was instrumental in navigating the complexities of persona simulation and keeping our work focused and innovative.

We also extend our sincere thanks to the faculty members of the Computer Science department for their academic support. Finally, our appreciation goes to our friends and colleagues who provided moral support and participated in our user testing phases, offering crucial feedback that helped shape the final product.

*(Page Break)*

Abstract

Users of modern conversational AI are increasingly seeking personalized, emotionally aware, and relatable digital companions. However, current chatbots often provide generic responses, failing to reflect a user’s unique communication style or personality. This project, "AI Mirror," addresses this gap by developing a novel system that creates an AI character designed to mirror the user’s persona. The primary objective is to move beyond simple role-playing and achieve a deep, personalized simulation based on the user's own conversational data.

The methodology evolved through several experiments, moving from initial fine-tuning attempts with models like Phi-3 using LoRA, which proved to be resource-intensive and led to persona confusion. The final, successful architecture employs a Retrieval-Augmented Generation (RAG) pipeline built upon a powerful pre-trained Large Language Model (Llama 3.3 70B via Groq), guided by sophisticated prompt engineering. The system supports multi-modal data ingestion, processing user-provided chat logs from WhatsApp and Telegram, voice recordings transcribed via OpenAI's Whisper, and responses from a structured questionnaire. This data is used to generate a detailed character profile, which is then vectorized into a FAISS store for efficient, context-aware memory retrieval. The entire system is integrated into a cross-platform React Native mobile application, featuring multi-modal interaction including text chat, voice messaging, and real-time voice calls using TTS technologies like gTTS and Coqui AI.

The results demonstrate that the final RAG-based approach provides fast, coherent, and contextually relevant responses that successfully capture the user's style, emotion, and persona. This work provides a robust framework for creating deeply personalized AI companions, significantly enhancing the potential for natural and meaningful human-AI interaction.

**Keywords:** Persona Simulation, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), Personalized Dialogue, Multi-Modal AI, Prompt Engineering, FAISS, Conversational AI, Voice Chatbots.

*(Page Break)*

Table of Contents

*(Auto-generate this in your word processor)*

*(Page Break)*

List of Figures

*(Auto-generate this in your word processor)*

*(Page Break)*

List of Tables

*(Auto-generate this in your word processor)*

*(Page Break)*

Chapter 1: Introduction

1.1 Background

In recent years, the field of conversational Artificial Intelligence (AI) has witnessed exponential growth, largely driven by the advent of powerful Large Language Models (LLMs). These models have enabled the development of sophisticated chatbots and virtual companions capable of understanding and generating human-like text with remarkable fluency. As users become more accustomed to interacting with AI, their expectations have evolved beyond mere functional assistance. There is a growing demand for AI companions that are not just intelligent but are also personal, emotionally aware, and relatable. The next frontier in conversational AI lies in moving from generic, one-size-fits-all agents to deeply personalized entities that can understand, reflect, and adapt to an individual's unique identity.

1.2 Problem Statement

Despite their advanced capabilities, most contemporary chatbots and AI companions fail to capture the nuances of an individual's personality. They often rely on pre-scripted responses or a generic, pre-defined persona, resulting in conversations that feel monotonous, impersonal, and lack genuine connection. This lack of true personalization prevents the formation of a deep and meaningful bond between the user and the AI. The core problem this project addresses is **the inability of existing systems to create a high-fidelity AI simulation that truly mirrors a specific user's communication style, emotional tone, personal history, and unique personality based on their own historical data.**

1.3 Project Objectives

To address the stated problem, this project aims to achieve the following specific, measurable, achievable, relevant, and time-bound (SMART) objectives:

1. **To design and develop a multi-modal system** capable of creating a personalized AI character that mirrors a real user's personality.
2. **To implement a flexible data ingestion pipeline** that accepts user data from three distinct sources: a structured questionnaire, WhatsApp chat logs, and Telegram chat logs.
3. **To build a robust persona generation module** that uses an LLM to analyze the ingested data and automatically generate a detailed character profile, including a narrative biography and structured key-value answers.
4. **To implement a Retrieval-Augmented Generation (RAG) architecture** using a FAISS vector store to provide the AI character with a persistent, context-aware memory of the user's data.
5. **To integrate the AI system into a cross-platform mobile application** (React Native) that supports multi-modal interaction: text chat, asynchronous voice messaging, and real-time voice calls.
6. **To provide a simple and intuitive user experience** for creating and interacting with the AI persona, abstracting away all technical complexity from the end-user.

1.4 Project Scope

The scope of the AI Mirror project is defined as follows:

* **In-Scope:**
* The system creates a single, mirrored AI persona per user based on the data they provide.
* Data ingestion is supported via a manual questionnaire, WhatsApp chat import (via QR code), and a user-friendly Telegram login flow (via phone and code).
* The core AI methodology relies on inference and prompt engineering with a pre-trained LLM (Llama 3.3 70B), augmented by a RAG pipeline.
* The system supports multi-modal chat, including text, voice input (Whisper transcription), voice output (gTTS/Coqui TTS), and a dedicated voice call screen.
* An animated avatar is included to enhance the user's sense of interaction.
* **Out-of-Scope:**
* The system does not support the creation of multiple fictional characters. Its focus is solely on mirroring the user.
* Data is not automatically scraped from social media or other platforms; all data ingestion is user-initiated and requires consent.
* The project does not involve training or full fine-tuning of a base LLM from scratch due to computational constraints.
* The lip-syncing feature for the avatar is in a developmental stage and not fully implemented.

1.5 Significance of the Project

The significance of AI Mirror lies in its novel approach to hyper-personalization in conversational AI. Unlike existing commercial applications like Character.ai or Replika, which focus on role-playing with fictional or pre-defined characters, our system pioneers the concept of creating a digital "mirror" of the user. This provides several key benefits:

* **Enhanced User Engagement:** By mimicking the user's own style, emotion, and history, the AI fosters a deeper, more natural, and more meaningful connection.
* **Therapeutic and Self-Reflective Potential:** The system can serve as a tool for self-reflection, allowing users to interact with a version of themselves, potentially leading to new insights and understanding.
* **Practical Application of SOTA Research:** It successfully implements state-of-the-art concepts like RAG and multi-modal interaction in a real-world mobile application, addressing a clear gap in the market for true user-centric persona simulation.

1.6 Report Structure

This report is organized into six chapters. Chapter 2 provides a review of the theoretical background and related work. Chapter 3 details the system analysis and design, including the architecture and design diagrams. Chapter 4 describes the implementation environment, tools, and challenges faced. Chapter 5 presents the system testing strategy and evaluation. Finally, Chapter 6 concludes the report with a summary, limitations, and recommendations for future work.

*(Page Break)*

Chapter 2: Literature Review & Theoretical Background

2.1 Theoretical Background

* **Large Language Models (LLMs):** LLMs are deep learning models with billions of parameters, pre-trained on vast amounts of text data. Our project ultimately utilized Meta's **Llama 3.3 70B-Versatile** model, accessed via the **Groq** inference engine, selected for its exceptional speed and strong reasoning capabilities, making it ideal for real-time conversational applications.
* **Retrieval-Augmented Generation (RAG):** RAG enhances LLM performance by grounding responses in external knowledge. It combines a **retriever** (fetching relevant information) with a **generator** (the LLM). In our system, the user's data is vectorized using **Sentence-Transformers (all-MiniLM-L6-v2)** and stored in a **FAISS (Facebook AI Similarity Search)** vector store. When the user sends a message, the RAG pipeline retrieves relevant information to provide context for the LLM, ensuring more accurate and personalized replies.
* **Fine-Tuning vs. Prompt Engineering:**
* **Fine-Tuning:** Our initial experiments involved fine-tuning models like Phi-3 using **LoRA (Low-Rank Adaptation)**. This proved to be resource-intensive, slow, and led to "persona confusion," where the model struggled to differentiate between multiple training personas.
* **Prompt Engineering:** Our final, successful methodology relies on this technique. We dynamically construct a detailed meta-prompt that includes the character's generated biography, Q&A, and recent chat history. This prompt "instructs" the pre-trained LLM on how to behave, achieving high-fidelity persona simulation without altering the model's weights.
* **Speech & Avatar Technologies:**
* **Speech-to-Text (STT):** We use **OpenAI's Whisper (Large V3 model)** to transcribe user voice recordings, chosen for its high accuracy and robust support for various Arabic dialects.
* **Text-to-Speech (TTS):** We utilize both **gTTS (Google Text-to-Speech)** for its simplicity and **Coqui AI's XTTS model** for its high-quality voice cloning capabilities, allowing the AI's responses to be spoken in a personalized voice.

2.2 Literature Review

Our work is built upon extensive research in persona-based dialogue systems.

* **Foundational Datasets:** The field was significantly advanced by datasets like **PersonaChat** (Facebook AI, 2018) and **ConvAI2**, which provided agents with explicit profiles to make conversations more personal. These highlighted the data scarcity problem that our system overcomes by using the user's own data.
* **Advanced Mechanisms for Persona Consistency:** The paper **"Personalized** Dialogue Generation with Persona-Adaptive **Attention" (PAA)** (2022) introduced a novel attention mechanism to dynamically balance a speaker's persona with dialogue context. This demonstrated that strong performance could be achieved with limited data, an insight that validated our move away from data-hungry fine-tuning methods.
* **Simulating Personas with Pre-trained LLMs:** More recent research, such as **"Characteristic AI Agents via Large Language Models"** (2024), has shifted focus from fine-tuning to guiding large, pre-trained models. Their framework relies on creating detailed character profiles and using them to engineer prompts, a methodology that is the primary inspiration for our system's core architecture. Their introduction of the **Character100** benchmark also highlighted the growing interest in evaluating the ability of LLMs to simulate real personalities.
* **Retrieval-Augmented Generation:** The seminal paper **"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks"** (Lewis et al., 2020) proposed the RAG framework that we have adopted. By combining dense retrieval with a sequence-to-sequence model, they demonstrated higher factual accuracy and relevance, a principle we apply to ground our AI's responses in the user's personal "knowledge base."

**Research Gap:** While the reviewed literature shows significant progress in creating agents with *pre-defined* or *fictional* personas, a clear gap exists in creating high-fidelity agents that can dynamically **mirror a real, living user's personality** based on their own complex and multi-modal conversational history, including real-time voice interaction. Our project, "AI Mirror," directly addresses this gap.

*(Page Break)*

Chapter 3: System Analysis and Design

3.1 Development Methodology

The project was developed using an **Iterative and Agile-inspired methodology**. Given the experimental nature of AI model integration and the evolving understanding of the project requirements, a rigid Waterfall model was unsuitable. Our process involved a series of cycles:

1. **Experimentation:** Each cycle began with an experiment (e.g., fine-tuning with LoRA).
2. **Evaluation:** We assessed the results of each experiment against our objectives.
3. Adaptation: Based on the evaluation and supervisor feedback, we adapted our approach, pivoting from fine-tuning to a RAG + Prompt Engineering architecture after early experiments proved inefficient.  
   This iterative approach allowed for flexibility and the gradual refinement of the system.

3.2 Requirements Gathering

* **Competitive Analysis:** We analyzed applications like Character.ai and Replika to define our unique value proposition (mirroring a real user).
* **Supervisor Meetings:** Regular discussions helped clarify academic requirements and refine the project's technical direction.
* **Prototyping:** Building and testing different versions served as a primary method for discovering technical constraints and user experience needs.

3.3 Requirements Analysis

3.3.1 Functional Requirements (FR)

* **FR1:** The system shall allow a user to create a persona via a structured questionnaire.
* **FR2:** The system shall allow a user to import a WhatsApp chat log to generate a persona.
* **FR3:** The system shall provide a secure and user-friendly login flow for Telegram to import a chat log.
* **FR4:** The system shall process imported data to automatically generate a character biography and key-value answers using an LLM.
* **FR5:** The system shall store the generated character profile persistently on the backend and locally on the client device.
* **FR6:** The system shall allow the user to engage in a text-based conversation with their AI persona.
* **FR7:** The system shall support asynchronous voice messaging (record, send, transcribe, reply).
* **FR8:** The system shall support a real-time voice call mode (voice-in, voice-out).
* **FR9:** The system shall use a RAG mechanism to retrieve context from the user's data to inform responses.
* **FR10:** The system shall synthesize the AI persona's text replies into speech for voice messages and calls.

3.3.2 Non-Functional Requirements (NFR)

* **NFR1 (Performance):** The AI's response time in text chat should be near real-time (under 5 seconds). Voice call latency should be minimized for a natural feel.
* **NFR2 (Usability):** Persona creation flows must be intuitive and require no technical expertise.
* **NFR3 (Security):** User data and session tokens must be handled securely and not be stored unencrypted.
* **NFR4 (Reliability):** The system must gracefully handle failures in external API calls and provide clear error messages.

*(Page Break)*

3.4 System Design

This section provides a detailed breakdown of the system's design through a series of UML diagrams, illustrating the architecture, user interactions, and data flow.

3.4.1 System Architecture Diagram

The architecture of AI Mirror is a distributed system composed of three primary components: the mobile frontend, a specialized middleware for WhatsApp, and the main AI backend. This design ensures modularity and scalability.

@startuml  
!theme vibrant  
title AI Mirror - System Architecture  
  
cloud "External Services" {  
    [Telegram API]  
    [WhatsApp Web]  
    [Groq API (LLM)]  
    [AI/NLP Models\n(Whisper, TTS)]  
}  
  
node "User's Device" {  
    frame "Mobile App (React Native)" as App {  
        [UI Components\n(Screens, Buttons)]  
        [Local Storage\n(AsyncStorage)]  
        [API Client (fetch/axios)]  
    }  
}  
  
node "Middleware Server (for WhatsApp)" {  
    frame "Node.js Server" as NodeServer {  
        [Express API]  
        component "Puppeteer\n(Headless Chrome)" as Puppeteer  
        [whatsapp-web.js]  
    }  
}  
  
node "Main AI Backend" {  
    frame "Python Server (FastAPI)" as PythonServer {  
        [FastAPI Endpoints\n(/chat, /call, /telegram/\*)]  
        [Telethon Client]  
        folder "AI Core" {  
            [RAG Pipeline]  
            [Sentence Transformer]  
        }  
        database "Character Data Storage" {  
            [FAISS Vector Store]  
            [JSON Profiles]  
        }  
    }  
}  
  
' --- Connections ---  
App::[API Client] ..> NodeServer::[Express API] : HTTP Request (WhatsApp Flow)  
App::[API Client] ..> PythonServer::[FastAPI] : HTTP Request (All other flows)  
  
NodeServer::[whatsapp-web.js] --> Puppeteer  
Puppeteer <--> [WhatsApp Web]  
  
PythonServer::[Telethon Client] <--> [Telegram API]  
PythonServer::[RAG Pipeline] ..> [Groq API (LLM)] : LLM Inference  
PythonServer::"AI & NLP Core" ..> [AI/NLP Models\n(Whisper, TTS)] : Model Loading/Inference  
PythonServer::[RAG Pipeline] <--> [FAISS Vector Store]  
  
@enduml

**Figure 3.1: High-Level System Architecture**

*(Page Break)*

3.4.2 Use Case Diagram

This diagram illustrates the main functionalities available to the user, including all persona creation methods and interaction modes.

@startuml  
!theme vibrant  
left to right direction  
  
actor User  
  
rectangle "AI Mirror System" {  
  usecase "Create Persona Manually\n(Questionnaire)" as UC1  
  usecase "Import Persona from WhatsApp" as UC2  
  usecase "Import Persona from Telegram" as UC3  
    
  usecase "Interact with Persona" as UC\_Interact  
    
  usecase "via Text Chat" as UC4  
  usecase "via Voice Message" as UC5  
  usecase "via Voice Call" as UC6  
}  
  
User --> UC1  
User --> UC2  
User --> UC3  
  
(UC1) ..> UC\_Interact : <<include>>  
(UC2) ..> UC\_Interact : <<include>>  
(UC3) ..> UC\_Interact : <<include>>  
  
UC\_Interact <|-- UC4  
UC\_Interact <|-- UC5  
UC\_Interact <|-- UC6  
@enduml

**Figure 3.2: System Use Case Diagram**

*(Page Break)*

3.4.3 Sequence Diagrams

**A. Questionnaire Persona Creation Flow:**

@startuml  
!theme vibrant  
title Sequence Diagram: Questionnaire Persona Creation  
  
autonumber  
actor User  
participant "App (React Native)" as App  
participant "Python Server (FastAPI)" as Server  
  
User -> App: Selects "Create New Character"  
App -> User: Navigates to QuestionnaireScreen  
User -> App: Fills out Bio and Answers questions  
User -> App: Clicks "Save & Continue"  
App -> Server: POST /store\_data (characterId, name, bio, answers)  
activate Server  
note over Server: Create profile.json, questionnaire.json,\nand generate/save FAISS index  
Server --> App: {status: "success"}  
deactivate Server  
App -> App: Save character locally (AsyncStorage)  
App -> User: Navigate to ChatScreen  
@enduml

**Figure 3.3: Sequence Diagram for Questionnaire Flow**

**B. WhatsApp Persona Import Flow:**

@startuml  
!theme vibrant  
title Sequence Diagram: WhatsApp Persona Import  
  
autonumber  
actor User  
participant "App (React Native)" as App  
participant "Node.js Server" as NodeServer  
participant "Python Server (FastAPI)" as PythonServer  
  
group WhatsApp Linking & Chat Selection  
    User -> App: Selects "Import from WhatsApp"  
    App -> NodeServer: GET /initiate-whatsapp  
    activate NodeServer  
    NodeServer -> User: Displays QR Code in terminal  
    User -> App: Scans QR with phone  
    NodeServer -> NodeServer: on('ready'), fetch chats & picUrls  
    deactivate NodeServer  
    loop Poll Status until "ready"  
        App -> NodeServer: GET /initiate-whatsapp  
    end  
    App -> NodeServer: GET /get-chats  
    NodeServer --> App: {chats: [...]}  
    User -> App: Selects a chat  
end  
  
group Message Fetching & AI Processing  
    App -> NodeServer: POST /fetch-chat-messages (chatId)  
    activate NodeServer  
    NodeServer -> NodeServer: client.getChatById(chatId)  
    NodeServer --> App: {messages: [...], characterName: "..."}  
    deactivate NodeServer  
      
    App -> PythonServer: POST /preprocessing\_whats\_data (messages, name, id)  
    activate PythonServer  
    note over PythonServer: Analyze text, generate Bio & Answers\nusing Groq API (LLM). Then save profile & FAISS index.  
    PythonServer --> App: {processedData: {bio, answers}}  
    deactivate PythonServer  
end  
  
group Finalization  
    App -> App: Save Character Profile locally  
    App -> User: Navigate to ChatScreen  
end  
@enduml

**Figure 3.4: Sequence Diagram for WhatsApp Import Flow**

**C. Voice Call Interaction Flow:**

@startuml  
!theme vibrant  
title Sequence Diagram: Voice Call Interaction  
  
autonumber  
actor User  
participant "CallScreen (App)" as App  
participant "Python Server (FastAPI)" as Server  
  
group User Speaks  
    User -> App: Holds mic button & speaks  
    App -> App: Records audio to local URI  
    User -> App: Releases button  
end  
  
group Backend Processing  
    App -> Server: POST /call (characterId, audio\_file)  
    activate Server  
    note over Server: 1. Transcribe audio using Whisper\n2. RAG Search in FAISS memory\n3. Construct Prompt & Call LLM (Groq)\n4. Synthesize LLM reply to audio with TTS  
    Server --> App: return FileResponse(audio\_file)  
    deactivate Server  
end  
  
group Bot Responds  
    App -> App: Receives audio file as blob  
    App -> App: Saves blob and plays it  
    User -> User: Hears voice response  
end  
@enduml

**Figure 3.5: Sequence Diagram for Voice Call Interaction**

3.4.5 Data Storage Design (ERD-like)

@startuml  
!theme vibrant  
title Character Data Structure  
  
entity "CHARACTER" as Char {  
  \* \*\*characterId\*\* (PK) : string  
  --  
  \* characterName : string  
  \* bio : text  
  \* imageUri : string  
}  
  
entity "QUESTIONNAIRE" as Q {  
  \* \*\*characterId\*\* (FK) : string  
  --  
  \* answers : json  
}  
  
entity "MEMORY\_STORE" as Mem {  
  \* \*\*characterId\*\* (FK) : string  
  --  
  \* faiss\_index : file  
  \* texts\_json : file  
}  
  
Char ||--|{ Q : "has"  
Char ||--|{ Mem : "has"  
  
note right of Char : "Core profile info."  
note right of Mem : "Vectorized memory for RAG."  
@enduml

**Figure 3.6: Logical Data Storage Structure**

*(Page Break)*

Chapter 4: Implementation

4.1 **Environment and** Tools

The AI Mirror system was developed using a modern, multi-platform technology stack.

* **Backend (Main Server):** Python 3.11, FastAPI, Uvicorn, Telethon, OpenAI Whisper, Coqui TTS, gTTS, Sentence-Transformers, FAISS, HTTpx.
* **Backend (Middleware for WhatsApp):** Node.js, Express.js, whatsapp-web.js (with Puppeteer).
* **Frontend (Mobile App):** React Native, Expo, React Navigation, Expo AV, Lottie, Axios.
* **Development & Deployment:** VS Code, Git, GitHub, ngrok.

4.2 System Screenshots

*(This section must be filled with actual screenshots from your running application.)*

(Placeholder for Figure 4.1: Character Selection Screen)

Caption: The main screen providing the three paths for persona creation: Questionnaire, WhatsApp, and Telegram.

(Placeholder for Figure 4.2: Telegram Login Screen)

Caption: The simple, step-by-step login process for Telegram.

(Placeholder for Figure 4.3: Chat Screen Interface)

Caption: The main chat interface showing a conversation, with support for both text and voice message input.

(Placeholder for Figure 4.4: Voice Call Screen Interface)

Caption: The dedicated voice call screen with a large "press-to-talk" button and visual feedback.

4.3 Implementation Challenges and Solutions

* **Challenge 1: Inefficiency of Fine-Tuning:** Our initial approach to fine-tune an LLM proved to be resource-intensive and led to "persona confusion."
* **Solution:** We pivoted to a RAG + Prompt Engineering architecture, which provided better and more consistent results without the need for training.
* **Challenge 2: Unreliable Inter-Server Communication:** Initial designs had the Node.js server calling the Python server via ngrok, which frequently resulted in ECONNRESET errors.
* **Solution:** We re-architected the data flow so the mobile app orchestrates the process, eliminating the unstable server-to-server link.
* **Challenge 3: Telegram Entity Not Found Error:** The telethon library would throw a Could not find the input entity error in a new session.
* **Solution:** We "warmed up" the session cache by calling client.get\_dialogs() before processing a specific chat, ensuring the client recognized all user entities.

*(Page Break)*

Chapter 5: System Testing and Evaluation

5.1 Testing Strategy

Our strategy involved Unit, Integration, End-to-End, and User Acceptance Testing (UAT) to ensure functionality, reliability, and usability across all features.

5.2 Test Cases

The following table provides a sample of test cases executed.

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Description** | **Status** |
| **TC-TG-01** | Successful Telegram Persona Import | **Pass** |
| **TC-TG-02** | Failed Telegram Login (Invalid Code) | **Pass** |
| **TC-WA-01** | Successful WhatsApp Persona Creation | **Pass** |
| **TC-Q-01** | Successful Questionnaire Persona Creation | **Pass** |
| **TC-CHAT-01** | Test Basic Text Chat Response | **Pass** |
| **TC-CHAT-02** | Test RAG Memory Retrieval | **Pass** |
| **TC-VOICE-01** | Test Asynchronous Voice Message | **Pass** |
| **TC-CALL-01** | Test Real-time Voice Call | **Pass** |

5.3 System Evaluation

The final system was evaluated against the objectives set in Chapter 1.

|  |  |  |
| --- | --- | --- |
| **Objective** | **Evaluation Method** | **Result** |
| **Create a personalized AI persona** | Qualitative review of generated bios and chat responses. | **Achieved.** |
| **Multi-modal data ingestion** | Testing all three import flows. | **Achieved.** |
| **Automatic profile generation** | Verification of server-side data files. | **Achieved.** |
| **Implement RAG with FAISS memory** | Executing TC-CHAT-02 and observing server logs. | **Achieved.** |
| **Integrate multi-modal mobile app** | Successful operation of text, voice message, and call features. | **Achieved.** |
| **Provide an intuitive user experience** | Feedback from UAT. | **Achieved.** |

*(Page Break)*

Chapter 6: Conclusion and Future Work

6.1 Conclusion

This project, "AI Mirror," successfully designed and implemented a novel system for creating deeply personalized AI companions. By evolving from resource-intensive fine-tuning to an efficient RAG and prompt engineering architecture, we overcame significant technical hurdles to deliver a robust and feature-rich application. The system's ability to ingest data from multiple sources (Questionnaire, WhatsApp, Telegram) and support multi-modal interactions (text, voice message, voice call) demonstrates a comprehensive solution to the problem of generic, impersonal chatbots. The final product meets all its core objectives, providing a unique framework for creating a digital "mirror" of a user that is both engaging and contextually aware.

6.2 Project Limitations

* **Dependency on External Services:** The system is reliant on the availability of Groq, OpenAI, Coqui, and messaging platform APIs.
* **WhatsApp Middleware Fragility:** The use of whatsapp-web.js relies on reverse-engineering and can be unstable.
* **Data Volume for Persona Quality:** The richness of the persona is directly proportional to the quality and quantity of the input data.
* **Static Memory:** The current RAG implementation does not dynamically update the memory with new interactions.

6.3 Future Work and Recommendations

* **Dynamic Memory Enhancement:** Implement a mechanism to continuously update the FAISS vector store with new interactions.
* **Expand Data Integration:** Add support for more data sources like Facebook Messenger or email.
* **Improve Avatar Interaction:** Fully implement lip-syncing for the animated avatar.
* **Experiment with On-Device Models:** Explore running parts of the inference process on-device to enhance privacy and reduce API dependency.
* **Conduct Formal User Studies:** Perform formal user studies to quantitatively measure user engagement and the perceived accuracy of the persona simulation.

*(Page Break)*

References

*(Format* these according to your university's required *style, e.g., IEEE or APA)*

1. Lewis, P., Perez, E., Piktus, A., et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. In Advances in Neural Information Processing Systems 33 (NeurIPS 2020).
2. Zhang, M., et al. (2018). *Personalizing Dialogue Agents: I have a dog, do you have pets too?*. arXiv preprint arXiv:1801.07243.
3. Song, H., et al. (2022). *Personalized* Dialogue Generation with Persona-Adaptive Attention. arXiv preprint arXiv:2210.15088.
4. Zhou, W., Wang, Y., Zhang, Y., et al. (2024). *Characteristic AI Agents via Large Language Models*. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.
5. Thoppilan, R., et al. (2022). *LaMDA: Language Models for Dialog Applications*.