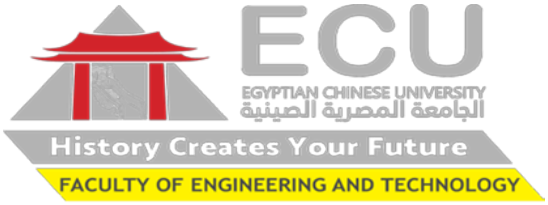


Faculty of Engineering and Information Technology

Mahmoud Essam Fathy 191900203

Hader Farag 191900162

Mariem Kamel 191900053

Mohamed Khaled 191900260

Alaa Kamal 191900057

Graduation Project Report

Wanas

December 2023

Under SuperVision of Dr. Amr Zamil

Abstract

This graduation project delves into the realm of mental health support by introducing an innovative application designed to act as a virtual therapist. In a world where the pace of life is relentless and mental well-being is of paramount importance, our project aims to leverage artificial intelligence and machine learning to provide accessible and personalized assistance. The virtual therapist application offers users a confidential and empathetic space to express their thoughts and emotions, fostering a sense of connection and understanding. This documentation explores the design principles, underlying technologies, and ethical considerations that guided the development of the application. Additionally, it delves into the potential impact of such a solution on mental health care, addressing the pressing need for accessible and stigma-free support in today's digital landscape. Join us on this journey as we navigate the intersection of technology and mental health, contributing to the ongoing dialogue about innovative solutions for well-being in the modern age.

Acknowledgements

We express deep gratitude for the collective support and contributions received during the completion of the graduation project. Special thanks are extended to academic mentors, advisors, peers, friends, family, participants, and the technological community. The acknowledgment emphasizes the collaborative nature of the project, recognizing the guidance, inspiration, and resources provided by various individuals and entities. It concludes with a sincere appreciation for everyone who played a role in the project's success and contributed to the academic and professional growth of the individuals involved. Wanas team

ECU

2023

Contents

1 Introduction **2**

1.1 Brainstorming and Idea Development 2

1.1.1 Initial Brainstorming Sessions 2

1.1.2 Decision-Making Process 2

1.2 Inspiration Behind WANAS 3

1.3 Introduction to WANAS 3

1.3.1 Project Overview 3

1.3.2 Purpose and Motivation 3

1.4 Understanding the Mental Health Crisis 4

1.4.1 Scope of the Problem 4

1.4.2 Consequences of Untreated Mental Health Issues 4

2 Related Work **7**

2.1 Introduction 7

2.2 Similar Apps 7

2.2.1 Talkspace 7

2.2.2 Woebot 8

2.2.3 Wysa 8

2.3 Functionality 8

2.3.1 Wysa 8

2.3.2 Woebot 9

2.3.3 Talkspace 10

2.4 Idea Inspiration 11

2.5 Comparison 11

2.6 Dependencies 11
2.7 Technical Related Work 13
2.8 Relevant Research Studies 13

3 Analysis And Design **15**

3.1 Functional Requirements 15
3.1.1 Login and Register 15
3.1.2 Chat with AI 16
3.1.3 Persona List 17
3.1.4 Chat List 17
3.1.5 Previous Conversations 17
3.1.6 Positive Card 18
3.1.7 Mood Tracker 18
3.2 Non-Functional Requirements 18
3.2.1 Security and Privacy 18
3.2.2 Speak Comfortably 19
3.2.3 Speed Of Response 19
3.2.4 Usability and User Experience 19
3.3 Implementation Strategies 19
3.3.1 Cross-Platform Development with Flutter 20
3.3.2 Advantages of Flutter 20
3.3.3 programming languages 21
3.3.4 Server-Side Programming with Go 21
3.3.5 AI and Machine Learning with Python 21
3.4 System analysis through formal diagrams 22
3.4.1 Wireframe 22
3.4.2 Use Case Diagram 23
3.4.3 Activity Diagram 24
3.4.4 Sequence Diagram 25
3.4.5 Class Diagram 26

4 AI **28**

4.1 Retrieval augmented generation (RAG) 28

4.1.1 Selecting Documents . . . 28
4.1.2 Chunking . . . 29
4.1.3 Similarity Search . . . 30
4.1.4 embedding in the prompt . . . 30
4.1.5 Introduction . . . 30
4.1.6 Selecting Dataset . . . 31
4.1.7 Preprocessing . . . 31
4.2 Data Splitting . . . 33
4.3 Selecting Model . . . 34
4.4 Fine-Tuning . . . 34
4.5 Training . . . 34
4.6 Result . . . 35
4.7 LoRA and Q-LoRA . . . 35
4.8 LoRA . . . 35
4.9 Q-LoRA . . . 36
4.10 GPU Usage . . . 36
4.11 Without Q-LoRA . . . 36
4.12 With Q-LoRA . . . 37
4.13 Simple Example . . . 37
4.14 Training Process . . . 37
4.15 Detail on Each Step . . . 38
4.16 Dataset . . . 38
4.17 Introduction . . . 38
4.18 Emotion Classification Model For Emotion Detection . . . 38
4.18.1 Preprocessing . . . 38
4.18.2 Text Separation . . . 38
4.18.3 Detailed Processing . . . 39
4.18.4 New DataFrame Creation . . . 39
4.18.5 Saving and Loading the Dataset . . . 39
4.19 Why Choosing GPT? . . . 39
4.20 Model Selection . . . 40
4.20.1 First Try: AceGPT7B . . . 40

4.20.2 Second Try: AceGPT13B-Chat 40
4.21 Final Model Choice 40

List of Figures

2.1 AraBERT 12

3.1 Login and Registration Flow Digram 16
3.2 Wireframe for the main screen of application 22
3.3 Use Case for application 23
3.4 Activity diagram for application 24
3.5 Sequence diagram for application 25
3.6 Class diagram for application 26

4.1 showing the chunking process and adding the chunks to the query 29
4.2 showing the chunking process and adding the chunks to the query 31

Chapter 1

Introduction

1.1 Brainstorming and Idea Development

1.1.1 Initial Brainstorming Sessions

As part of the WANAS project launch, numerous brainstorming sessions were held to generate innovative approaches to providing mental health care. Our group convened to discuss several concepts and determine the most effective strategy. These sessions were crucial in shaping the initial concept of WANAS, ensuring that every idea was considered and discussed.

1.1.2 Decision-Making Process

After generating a wide array of ideas, the next step was to evaluate them. The team assessed each idea based on several key criteria:

- **Feasibility:** Considering the practicality of implementing the idea within our available resources and timeframe.
- **Impact:** Evaluating how effectively the idea could address mental health needs and provide support to users.
- **Innovation:** Looking for unique and creative approaches that could set WANAS apart from other mental health support tools.

Through this structured decision-making process, we were able to narrow down our ideas and focus on the most promising concepts, laying a solid foundation for the development of WANAS.

1.2 Inspiration Behind WANAS

WANAS, named after the Arabic word brings the comforting essence of companionship and joy to the digital realm. In Egyptian culture, represents the warmth and comfort of being with loved ones, feeling understood and supported. Our application embodies this spirit, offering users a virtual therapist that provides empathetic, personalized mental health support, ensuring that no one feels alone in their journey towards better mental health.

1.3 Introduction to WANAS

1.3.1 Project Overview

WANAS is a virtual therapist application designed to provide personalized mental health support. Utilizing advanced artificial intelligence and machine learning, WANAS aims to deliver empathetic, context-aware interactions through a user-friendly interface. The application prioritizes user privacy and data security, ensuring a safe environment for users to express their thoughts and emotions.

1.3.2 Purpose and Motivation

Market Needs

The WANAS application aims to fill critical gaps in the current mental health care landscape. Despite this high prevalence, traditional mental health services often struggle with issues like accessibility, affordability, and stigma. In many areas, especially in low- and middle-income countries, there is a significant shortage of mental health professionals, which makes it challenging for individuals to get timely and adequate care. The growing need for mental health support, along with the shortcomings of existing services, highlights the necessity for innovative solutions like a virtual therapist application.

Academic Needs

From an academic perspective, the project aligns with the growing interest in leveraging artificial intelligence and machine learning for healthcare applications. It provides a practical implementation of AI and ML techniques in a socially impactful domain. The project offers a rich case study for exploring ethical considerations in AI, user interaction design, and the

development of adaptive systems. It also contributes to the body of research on digital mental health interventions, offering insights into their efficacy and user engagement.

1.4 Understanding the Mental Health Crisis

1.4.1 Scope of the Problem

Mental health issues are pervasive and affect people across all demographics worldwide. According to the World Health Organization (WHO), approximately 1 in 4 people will experience a mental health disorder at some point in their lives. This high prevalence reflects a wide range of conditions, from anxiety and depression to severe disorders like schizophrenia and bipolar disorder. Despite the widespread occurrence, there are significant challenges in addressing these issues effectively:

1. **Accessibility:** Many regions, particularly low- and middle-income countries, face a severe shortage of mental health professionals. This shortage makes it difficult for individuals to receive timely and adequate care.
2. **Affordability:** Traditional mental health services can be costly, making them inaccessible to large segments of the population, especially in regions without sufficient health insurance coverage or governmental support.
3. **Stigma:** Social stigma surrounding mental health issues often prevents individuals from seeking help. Fear of judgment and discrimination can lead to reluctance in accessing mental health services.
4. **Geographical Barriers:** In rural and remote areas, the availability of mental health services is often limited, requiring individuals to travel long distances to receive care.
5. **Timeliness:** Traditional mental health services typically involve scheduled appointments, which may not be available immediately when individuals are in crisis.

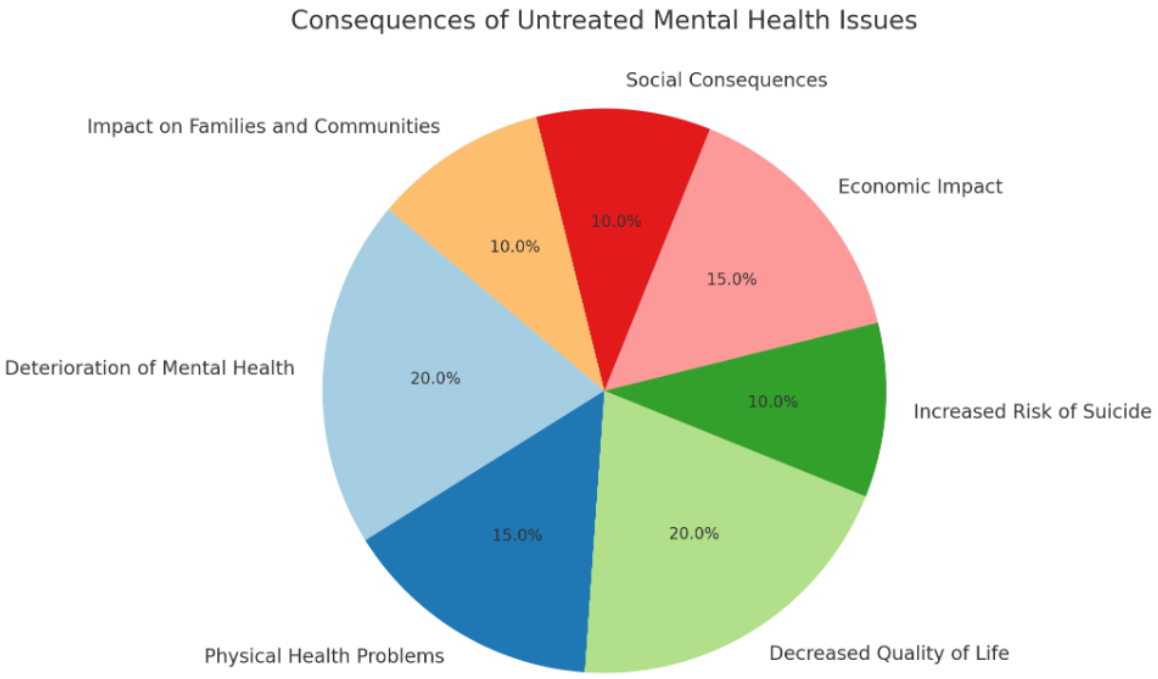
1.4.2 Consequences of Untreated Mental Health Issues

Untreated mental health issues have profound and multifaceted consequences, affecting individuals, families, communities, and society:

- **Deterioration of Mental Health:** Symptoms can worsen over time, complicating daily

life management.

- **Physical Health Problems:** Mental health issues can exacerbate chronic diseases and lead to unhealthy behaviors like substance abuse.
- **Decreased Quality of Life:** Affected individuals often struggle with relationships, employment, and enjoying activities.
- **Increased Risk of Suicide:** Conditions like depression and anxiety significantly heighten the risk of suicide.
- **Economic Impact:** Lost productivity, higher healthcare costs, and financial burdens on families and communities.
- **Social Consequences:** Stigma and isolation can result in social withdrawal and strained relationships.
- **Impact on Families and Communities:** Emotional distress, financial strain, and the challenge of providing support.



Chapter 2

Related Work

2.1 Introduction

In this chapter, we will briefly explore some prior works related to our project, focusing on mental health assistance applications and AI-driven chatbots designed for user interaction.

2.2 Similar Apps

Several applications bear similarities to ours. By examining these, we identified the necessary requirements and designed our application accordingly.

2.2.1 Talkspace

Talkspace provides therapy through text, audio, and video sessions with licensed therapists. It has made therapy more accessible, though it still faces challenges related to cost and availability. Talkspace offers:

- **Online Therapy**: Users can communicate with therapists via text, audio, and video messages.
- **Live Sessions**: Real-time therapy sessions through video calls.
- **Specialized Therapists**: Therapists who specialize in various mental health issues such as anxiety, depression, and PTSD.
- **Secure Messaging**: Ensures all communications are secure and confidential.

- **Comprehensive Resources**: Articles, worksheets, and self-help tools to support users between therapy sessions.

2.2.2 Woebot

Woebot is an AI chatbot offering cognitive-behavioral support by interacting with users, tracking their moods, and providing daily support. Key features include:

- **AI Chatbot**: Engages users in conversations using CBT techniques to provide coping strategies.
- **Mood Monitoring**: Tracks users' moods and provides feedback based on mood data.
- **Psychoeducation**: Educates users about mental health and CBT techniques.
- **Skill-Building**: Guides users through exercises to develop coping skills.
- **Personalization**: Tailors recommendations and resources to users' needs.

2.2.3 Wysa

Wysa is an AI chatbot that provides mental health support using techniques from cognitive-behavioral therapy (CBT), dialectical behavior therapy (DBT), and mindfulness. Its functionalities include:

- **AI Chatbot**: Empathetic conversations using natural language processing.
- **Self-Help Tools**: Tools and techniques from CBT, DBT, and mindfulness.
- **Mood Tracking**: Insights and patterns from tracking moods and emotions.
- **Personalized Support**: Tailored mental health resources and exercises.
- **Human Support**: Access to professional therapists if needed.

2.3 Functionality

2.3.1 Wysa

Wysa is an AI-driven chatbot designed to support mental health and well-being. It offers various functionalities to help users manage their mental health:

1. **AI Chatbot:**

- Provides empathetic conversations using AI to help users talk through their feelings.
- Uses natural language processing to understand and respond to users' concerns.

2. **Self-Help Tools:**
   - Offers tools and techniques based on Cognitive Behavioral Therapy (CBT), Dialectical Behavior Therapy (DBT), and mindfulness.
   - Provides exercises and activities to help manage stress, anxiety, and depression.
3. **Mood Tracking:**
   - Allows users to track their moods and emotions over time.
   - Provides insights and patterns based on mood tracking data.
4. **Personalized Support:**
   - Delivers personalized mental health resources and exercises tailored to users' needs.
   - Includes goal-setting and progress tracking features.
5. **Human Support:**
   - Offers access to professional therapists if users need human intervention.
   - Provides a blend of AI-driven support and human counseling.

2.3.2 Woebot

Woebot is an AI-powered mental health chatbot that uses principles of CBT to help users manage their mental health. Its functionalities include:

1. **AI Chatbot:**
   - Engages users in conversation to help them process their emotions and thoughts.
   - Uses CBT techniques to provide helpful responses and coping strategies.
2. **Mood Monitoring:**
   - Tracks users' moods and mental health status over time.
   - Provides feedback and suggestions based on mood data.
3. **Psychoeducation:**

- Educates users about mental health and CBT techniques.
- Offers informative content to help users understand their mental health better.

4. **Skill-Building:**
   - Guides users through exercises and activities to develop coping skills.
   - Uses evidence-based techniques to help users manage stress and anxiety.
5. **Personalization:**
   - Adapts to users' needs and preferences over time.
   - Provides tailored recommendations and resources.

2.3.3 Talkspace

Talkspace is an online therapy platform that connects users with licensed therapists. It offers several key functionalities:

1. **Online Therapy:**
   - Provides access to licensed therapists through text, video, and audio messages.
   - Allows users to communicate with their therapists asynchronously.
2. **Live Sessions:**
   - Offers live video therapy sessions with licensed therapists.
   - Enables real-time interaction and counseling.
3. **Specialized Therapists:**
   - Matches users with therapists who specialize in various mental health issues, such as anxiety, depression, and PTSD.
   - Allows users to select therapists based on their specific needs.
4. **Secure Messaging:**
   - Ensures all communications between users and therapists are secure and confidential.
   - Uses encryption to protect users' privacy.
5. **Comprehensive Resources:**

- Provides a range of mental health resources, including articles, worksheets, and self-help tools.
- Offers continuous support and guidance between therapy sessions.

2.4 Idea Inspiration

as going through the previous apps that were mentioned and many other apps we thought to give out something to the community that can contribute to benefit the mental health of individuals, as such we took a brief look on the Talkspace app and how it gives the user the benefit of contacting actual therapists and the enviroment of the app is healing then we decided to look further and then we found Woebot and Wysa that implement AI as a chat bot to chat with the user and make it more friendly and more comfy to not have embarrassment when talking to another human being.

2.5 Comparison

Other apps compared to ours we decided to add more features than just a chatbot to chat with:

- New personas every time the user wants to chat, enhancing comfort and engagement.
- A mood tracker that tracks daily mood and provides statistical insights over weeks, months, and years.
- Self-care features and daily challenges.
- Arabic language support for the chatbot, catering to Arabic-speaking users.

2.6 Dependencies

Our AI leverages existing models like BERT and AraBERT, fine-tuned to our requirements.

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art natural language processing model developed by Google that understands context in both directions in a sentence, significantly improving performance on a wide range of language tasks.

AraBERT is a pre-trained language model specifically designed for Arabic text, based on BERT's architecture. It addresses the unique challenges of Arabic text, including complex

Figure 2.1: AraBERT

morphology and diverse dialects. It is trained on large-scale Arabic corpora and fine-tuned for various NLP tasks, significantly improving the performance of Arabic NLP applications.

We also utilized datasets generated with ChatGPT for fine-tuning. now are currently using:

ACE GPT (Advanced Cognitive Engine for Generative Pre-trained Transformer)

ACE GPT is an advanced language model specifically designed to enhance natural language understanding and generation tasks. It builds upon the architecture of OpenAI's GPT models, integrating additional features and optimizations to improve performance in various applications, including conversational AI, text completion, and content generation. ACE GPT leverages large-scale datasets and fine-tuning techniques to provide contextually relevant and coherent responses, making it a powerful tool for developing intelligent chatbots and other AI-driven applications.

ACE GPT Chat

ACE GPT Chat is a specialized extension of ACE GPT focused on providing interactive conversational capabilities. It is tailored for real-time dialogue, offering users engaging and contextually appropriate responses. ACE GPT Chat incorporates advanced natural language processing techniques to maintain the flow of conversation, understand user intent, and deliver personalized support. This makes it an ideal solution for applications in customer service, virtual assistants, and mental health support chatbots.

2.7 Technical Related Work

Natural Language Processing (NLP) in Mental Health

NLP is crucial for creating AI-driven mental health applications, enabling AI to understand and generate human-like responses, analyze emotions, and offer empathetic support. Key NLP techniques include:

- **Sentiment Analysis**: This technique evaluates the emotional tone of user inputs, enabling AI to respond appropriately to various sentiments.
- **Emotion Detection**: It identifies specific emotions such as happiness, sadness, anger, and fear in user conversations, allowing AI to tailor its responses.
- **Conversational Agents**: These chatbots simulate human conversation, employing NLP techniques to understand queries, provide relevant responses, and engage in meaningful dialogues.
- **Text Classification**: This involves categorizing user inputs into predefined classes or topics, helping the AI to understand the context and intent behind the user's message.
- **Named Entity Recognition (NER)**: Identifying and classifying key information (names, dates, places, etc.) within the text to improve the AI's understanding of the conversation.
- **Dialogue Management**: Managing the flow of conversation to ensure coherent and contextually relevant interactions.

2.8 Relevant Research Studies

Several academic studies have explored AI and NLP in mental health, offering valuable insights:

- **AI Chatbots for Anxiety and Depression**: Studies show that AI chatbots can help reduce symptoms of anxiety and depression by using cognitive-behavioral techniques, though effectiveness varies with user engagement and AI response quality.
- **Sentiment Analysis in Mental Health**: Research indicates that sentiment analysis can accurately assess emotional states in text data, aiding mental health monitoring. However, ensuring accuracy and context-awareness remains challenging.
- **User Acceptance of AI-Driven Tools**: Studies suggest that users are generally open

to AI-driven mental health tools for initial support and self-help. However, data privacy, trust, and authenticity of AI responses are ongoing concerns.

- **Effectiveness of AI-Based Interventions**: Research has found varying levels of effectiveness for AI-based mental health interventions, often dependent on factors such as user engagement, the complexity of the mental health issue, and the quality of AI interactions.

Chapter 3

Analysis And Design

In this chapter, we outline the project planning process for the development of an AI-powered mental health application. We delineate the functional and non-functional requirements identified through extensive research, and detail the strategies employed to address these requirements.

3.1 Functional Requirements

Functional requirements delineate the specific features and capabilities that our application must possess to meet the needs and expectations of its users.

3.1.1 Login and Register

The application provides a user-friendly interface for new users to register and existing users to sign in.

User Registration If a user wants to register, they can do so in one of the following ways:

- **Creating a New Account:** Users can create a new account by providing necessary details such as username, password, email, and other required information.
- **Signing Up Using Google or Facebook:** Users have the option to sign up using their existing Google or Facebook accounts. This option simplifies the registration process by allowing users to authenticate using their social media credentials.

User Login Users who already have an account can sign in to the application by:

- **Standard Login:** Entering their registered username and password.
- **Social Media Login:** Logging in through their Google or Facebook accounts, provided they have previously registered using these credentials.

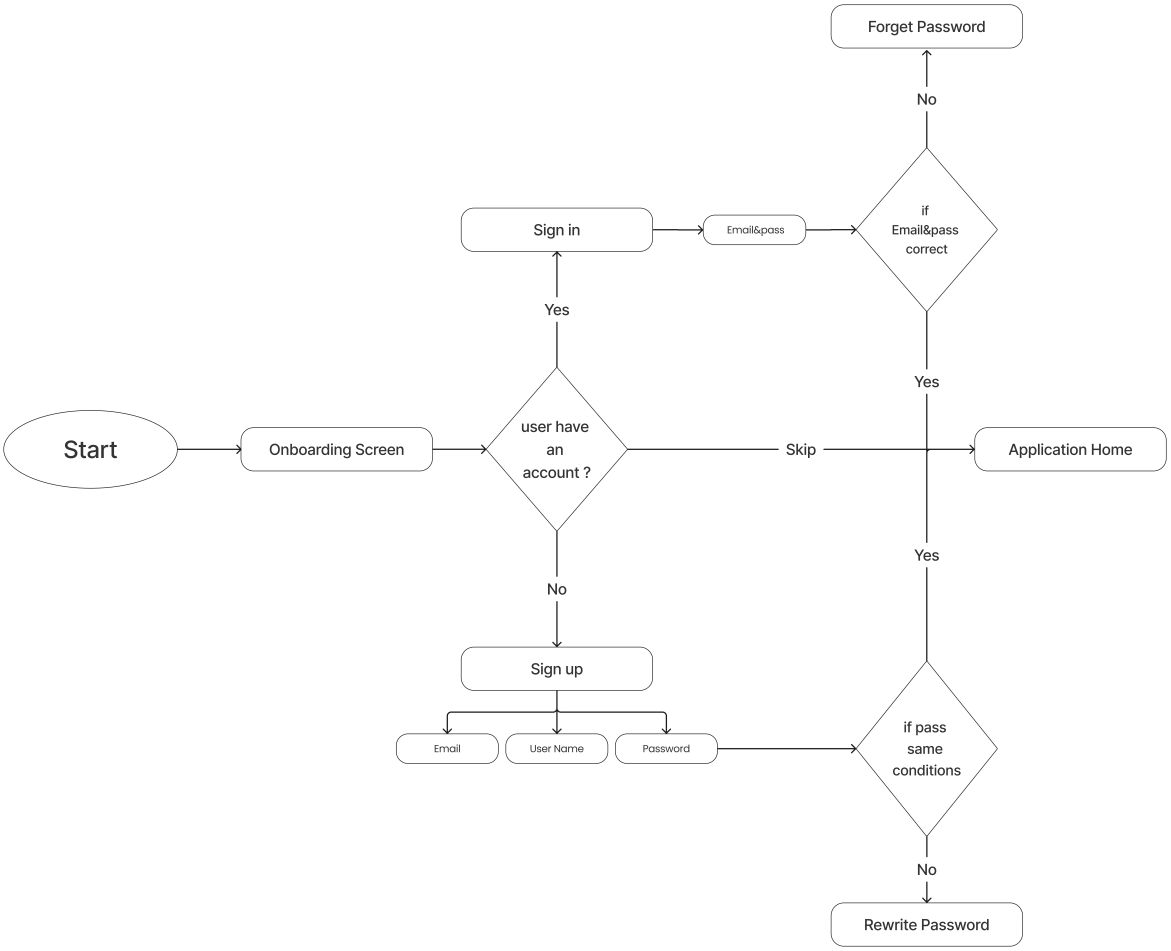


Figure 3.1: Login and Registration Flow Digram

3.1.2 Chat with AI

The application allows users to initiate and engage in chat sessions with the AI, enabling them to have meaningful conversations and seek mental health support.

Users can :

- **Starting a New Conversation:** Users can click on the "Start Chat" button to initiate a new conversation with the AI.

3.1.3 Persona List

The Persona List feature allows users to create and manage a list of characters with whom they can engage in conversations. The application saves all interactions with each character, providing a personalized experience.

Users can:

- **Create New Persona:** Add new personas by providing Persona details such as name.
- **Edit Existing Persona:** Modify details of existing personas to update their characteristics.
- **Delete Persona:** Remove Personas from the list if they are no longer needed.

3.1.4 Chat List

Within each persona, the Chat List feature provides a detailed list of conversations the user has had with that character. This organization helps users manage and review their interactions effectively.

Users can:

- **Select a Persona:** Choose a persona from the Persona List to view all associated conversations. ry in the Chat List includes the date and time of the conversation, providing a chronological overview.
- **Access Detailed Chats:** Click on an entry to view the full conversation, allowing users to revisit and analyze past interactions.

3.1.5 Previous Conversations

The application provides a feature for users to view their previous interactions, allowing them to track and reference past conversations with ease.

Users can:

- **View Details:** Each conversation entry includes the name of the interaction, duration, and date. This provides users with a quick overview of their past chats.

- **Select and Review:** Users can select a conversation to review the chat history, facilitating continuity and reference to past discussions.

3.1.6 Positive Card

The application includes an interactive *Positive Card* feature designed to enhance the user's mood and foster positivity.

Functionality of the Positive Card The Positive Card provides:

- **Interactive Content:** Engaging content such as motivational quotes, positive affirmations, or uplifting messages tailored to improve the user's mood.

3.1.7 Mood Tracker

The Mood Tracker feature allows users to monitor their emotional state over time, helping them identify their emotions in the period of using the app and effectively manage their mental health.

Users can:

- **Log emotional states:** Record your mood at different times throughout the day or week, or choose from predefined categories.
- **View Mood History:** Access a historical record of your recorded moods, presented in an intuitive interface that includes graphs or charts showing mood trends over time.

3.2 Non-Functional Requirements

Non-functional requirements encompass the attributes that characterize the overall performance, usability, and scalability of our application.

3.2.1 Security and Privacy

Ensuring the security and privacy of user data is paramount in our application. We implement robust security measures to maintain the confidentiality of users and ensure complete privacy. Data protection is enforced through secure communication channels and encryption between the server, the user, and the AI model.

Data Protection

- **Encryption:** All user data is encrypted during transmission and storage to prevent unauthorized access.
- **Secure Authentication:** Implement secure authentication mechanisms such as multi-factor authentication to verify user identity.
- **Compliance:** Adhere to privacy regulations and standards to ensure user data is handled responsibly and legally.

3.2.2 Speak Comfortably

Our application supports the Arabic language, enabling users to communicate comfortably and express themselves accurately in their native language. This language support is crucial for effective interaction and user satisfaction.

3.2.3 Speed Of Response

our application is designed to provide rapid responses to user inputs, facilitating a smooth and continuous conversation experience without long wait times.

3.2.4 Usability and User Experience

Our application features a user-friendly interface that is simple and easy to navigate, allowing users to interact with the application effortlessly.

User-Centric Design

- **Intuitive Interface:** We Designed an interface that is easy to learn and use, with clear navigation and well-organized elements.
- **Aesthetic Appeal:**we Ensured the visual design is appealing and enhances the user experience.

3.3 Implementation Strategies

To address the identified functional and non-functional requirements, various implementation strategies and technologies are employed in the development of the application.

3.3.1 Cross-Platform Development with Flutter

Utilizing Flutter for cross-platform development enables applications to reach a broader audience across different devices and operating systems, enhancing accessibility and user engagement. Flutter's unique approach to building natively compiled applications for mobile, web, and desktop from a single codebase ensures that the application can cater to a diverse user base without needing separate versions for each platform. This strategy saves significant time and resources [**flutter˙overview**].

3.3.2 Advantages of Flutter

Broader Accessibility Flutter allows the application to be compatible with various operating systems, including iOS, Android, Windows, macOS, Linux, and Web. This wide range of support ensures that users can access the application regardless of their preferred device or platform .

Consistent User Experience Flutter provides a rich set of customizable widgets and its own rendering engine, enabling the creation of a consistent user experience across different devices. This consistency is crucial for delivering a seamless and unified user interface and interaction flow, which enhances user satisfaction and engagement. The ability to achieve pixel-perfect UI on various platforms ensures that users enjoy a similar experience, irrespective of the device they use [**flutter˙cons˙pros**].

Cost-Efficiency Flutter's approach to a single codebase for multiple platforms significantly reduces development costs compared to creating and maintaining separate codebases for each platform. This cost-efficiency extends to updates and maintenance since changes can be implemented once and reflected across all platforms, eliminating the need for redundant development efforts [**codecademy˙flutter**].

Faster Time-to-Market Flutter accelerates the deployment process by allowing the development of a single codebase that can be compiled into applications for multiple platforms. This rapid development and deployment capability are crucial for addressing user needs promptly and maintaining competitiveness in the market. Flutter's hot-reload feature further speeds up the development process by enabling real-time changes without restarting the app [**flutter˙hot˙reload**].

Simplified Testing With a unified codebase in Flutter, the testing process is streamlined, allowing developers to identify and address issues more efficiently. This simplification ensures higher quality and reliability of the application across all supported platforms. Additionally, Flutter's comprehensive testing framework supports unit, widget, and integration testing, facilitating thorough testing processes [**flutter˙testing**].

3.3.3 programming languages

The choice of programming languages plays a crucial role in the development of the application. Go is utilized for server-side programming due to its efficiency, concurrency support, and scalability, while Python is leveraged for artificial intelligence programming owing to its extensive libraries and frameworks tailored for machine learning and natural language processing tasks.

3.3.4 Server-Side Programming with Go

Efficiency and Performance: Go (Golang) offers excellent performance due to its compiled nature and efficient garbage collection. This performance is crucial for handling large volumes of requests and ensuring smooth operation of the application [**golang˙efficiency**].

Concurrency Support: Go's native support for concurrency through goroutines allows the application to manage multiple processes simultaneously, enhancing the system's ability to handle concurrent user interactions and background tasks [**golang˙concurrency**].

Scalability: Go's design principles and simplicity make it well-suited for building scalable server-side applications, ensuring the AI mental health application can grow and adapt to increasing user demands [**golang˙scalability**].

3.3.5 AI and Machine Learning with Python

Extensive Libraries: Python is chosen for artificial intelligence programming due to its rich ecosystem of libraries and frameworks, such as TensorFlow, Keras, and scikit-learn. These tools provide robust support for developing and deploying AI models, including those for natural language processing and sentiment analysis [**tensorflow**, **keras**, **scikit˙learn**].

Ease of Integration: Python's flexibility and ease of integration with other technologies facilitate the incorporation of AI capabilities into the application, enabling the development of

sophisticated mental health support features [**python˙integration**].

3.4 System analysis through formal diagrams

The next section provides a detailed analysis of the system through Four formal diagrams and a wireframe. These diagrams will delve into different aspects of the system, including an outline of what the application looks like, users interacting with it (Use Cases), its internal processes (activities), and the flow of communications between components (sequences). In addition, they will identify the basic elements (Class Digram) of the system and its different operational states.

3.4.1 Wireframe

In this wireframe, we present the initial design of the application and present the most important requirements that make up the application

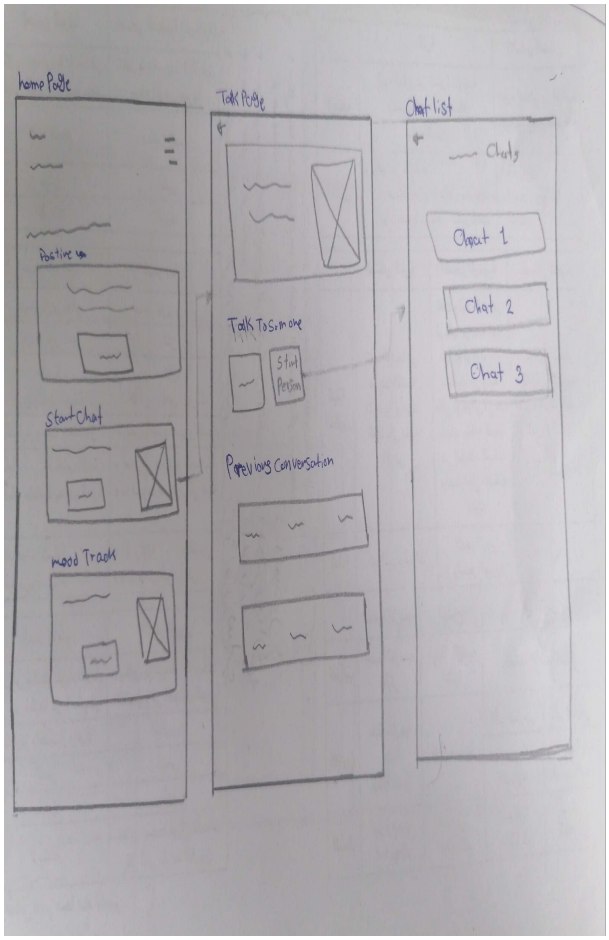


Figure 3.2: Wireframe for the main screen of application

3.4.2 Use Case Diagram

The use case diagram depicts an app "WANAS," with the user as the central actor. It outlines the actions users can perform within the system, such as registering, logging in, editing their profiles, starting a chat, interacting with personas, tracking their mood, accessing tips, deleting their account, and logging out. There are also extended use cases, including editing passwords, email, and pictures, suggesting a focus on user management and personalization.

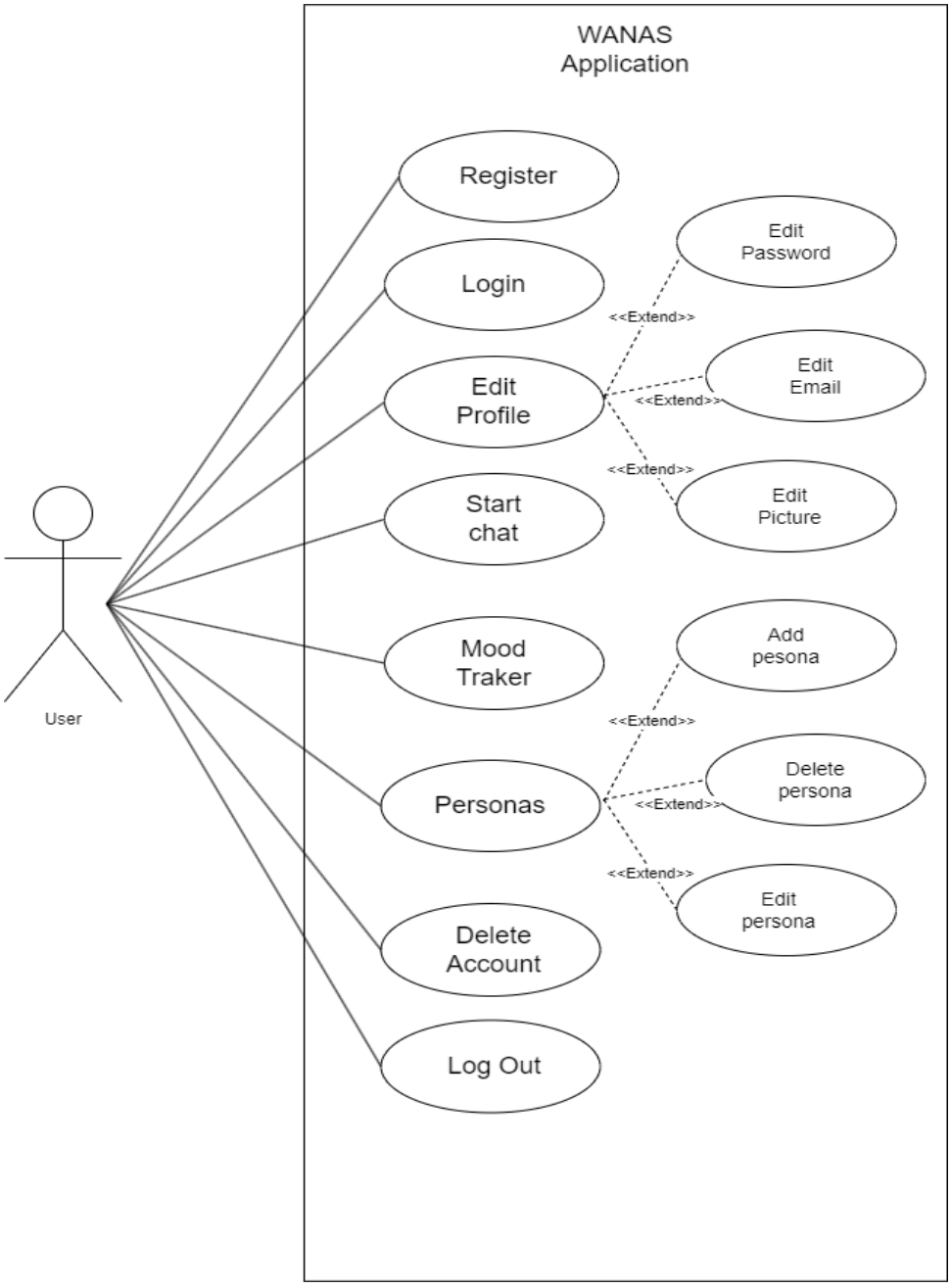


Figure 3.3: Use Case for application

3.4.3 Activity Diagram

The flow of actions that the user can perform is depicted in the following figure:

- It displays the flow of the Login and Register processes. The user can select to reach the home page by registering to create a new account or by logging in if he already has one. Without validating, the user won't be able to view the home page.
- After logging in user can make changes in his profile , start chat , personas list , choose mood.

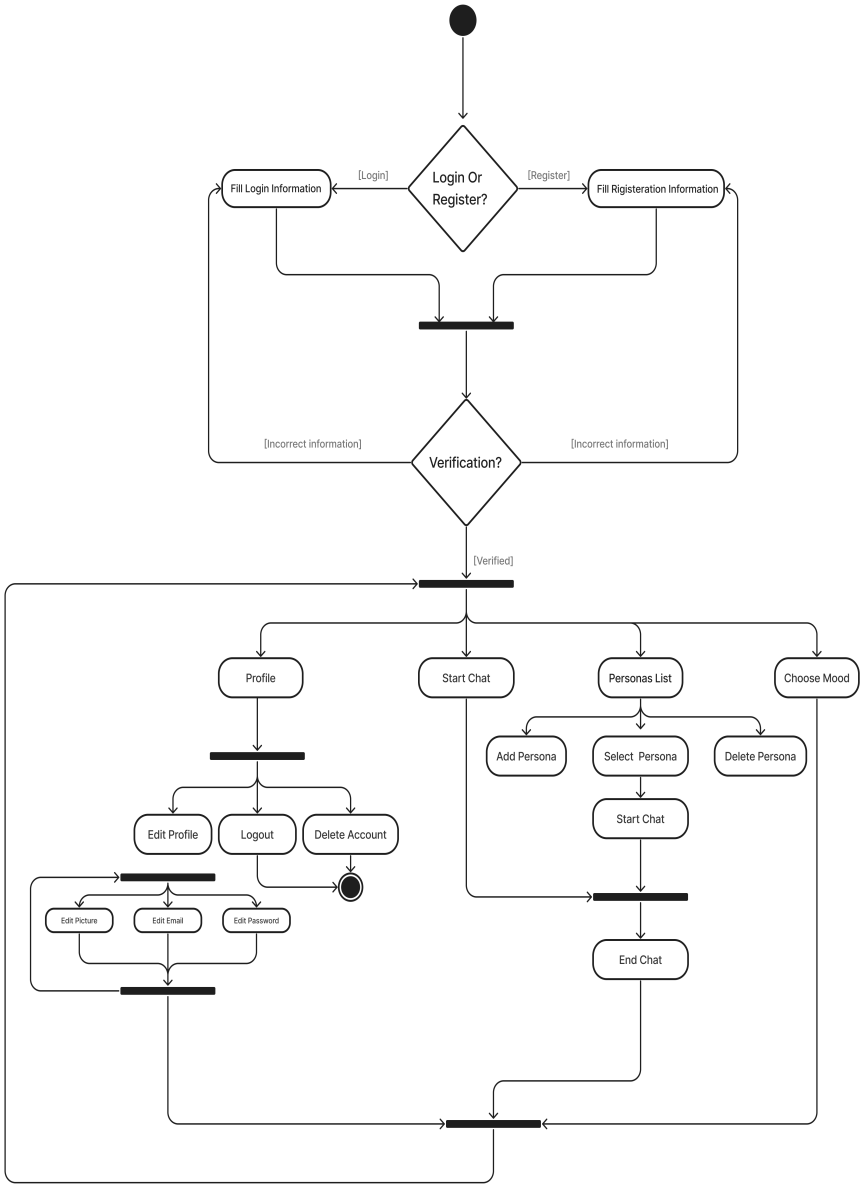


Figure 3.4: Activity diagram for application

3.4.4 Sequence Diagram

The sequence diagram shows the interactions between a user and a system in a chat application. The user starts by logging in or registering. After successful authentication, the user can create a chat, choose a state, edit the state, edit their profile picture, edit their email, or edit their password. The system interacts with the database to save and retrieve user data and chat information. The mood tracker and the profile features are also represented in the diagram.

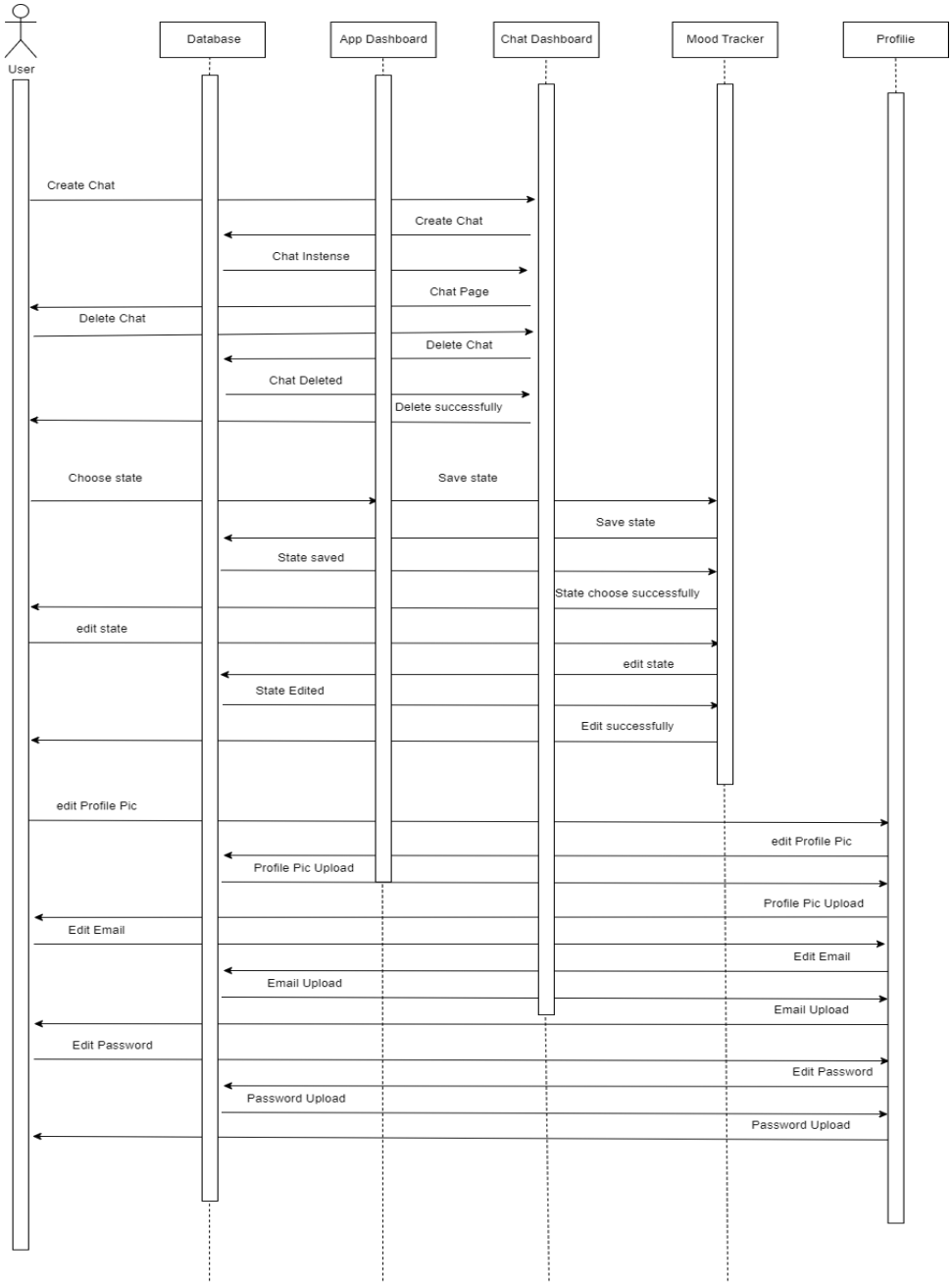


Figure 3.5: Sequence diagram for application

3.4.5 Class Diagram

The following figures represents the functions and attributes of the four main classes that the system use :

- Persona .
- User.
- Message .
- Chatbot.

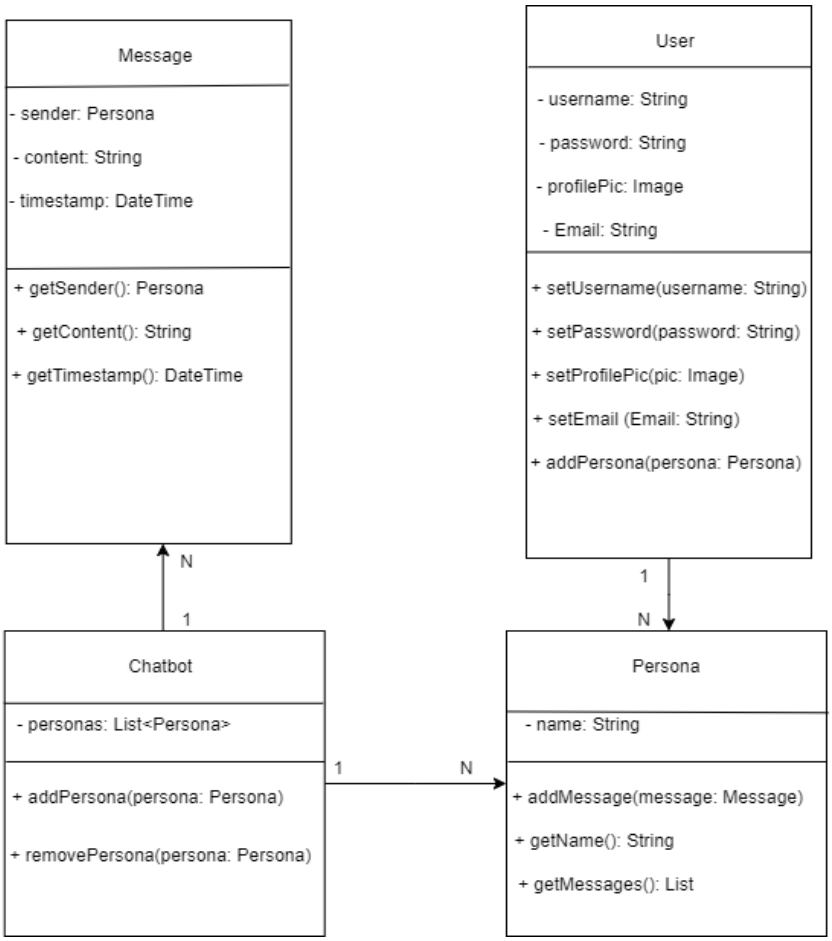


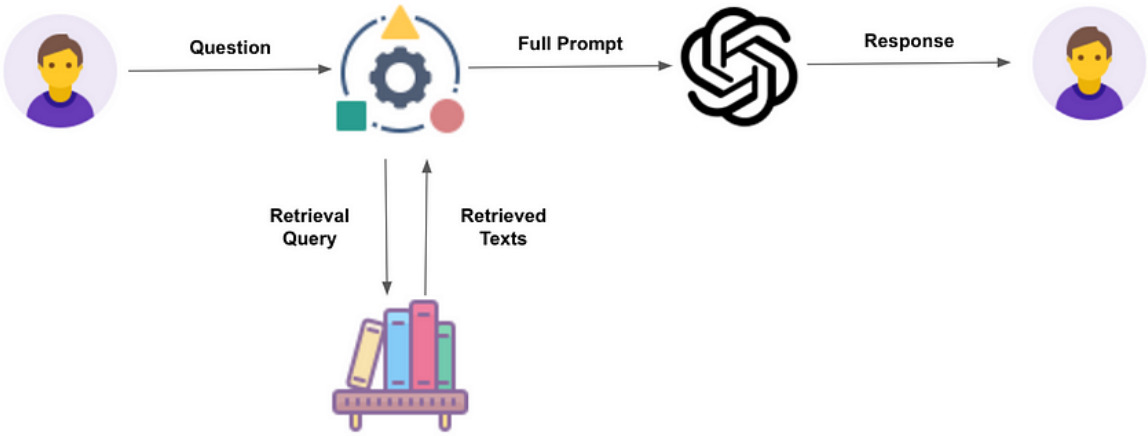
Figure 3.6: Class diagram for application

Chapter 4

AI

4.1 Retrieval augmented generation (RAG)

Retrieval augmented generation (RAG) is a natural language processing (NLP) technique that combines the strengths of both retrieval- and generative-based artificial intelligence (AI) models



why?

RAG allows the LLM to present accurate information with source attribution

4.1.1 Selecting Documents

First we have select an infromative documents to make the bot therapiest more informative and take the right decision towards the correct and scientific response towards the solution. so

we collected alot of arabic base and translated books and documents from scholar we need to collect alot of information so we made a web scraper to take inputs and search in google scholar and get back the top articles and books in pdf to store them for the next step to use them in the rag system to cover alot of points in the theraptic direction

4.1.2 Chunking

Dividing a large text corpus into smaller, manageable pieces or segments. Each chunk acts as a standalone unit of information that can be individually indexed and retrieved.

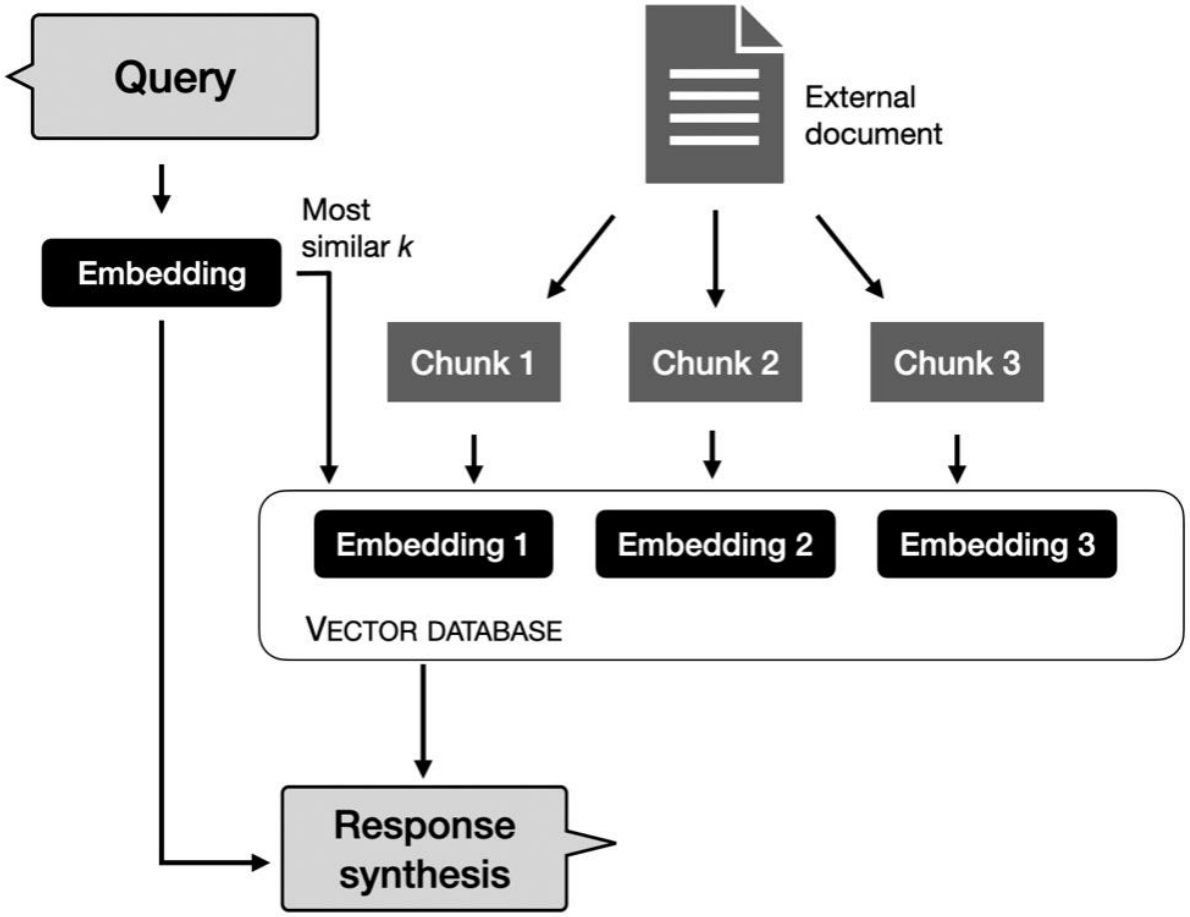


Figure 4.1: showing the chunking process and adding the chunks to the query

Chunking Methods

We considered several chunking methods for processing our PDF text:

- Fixed Size Chunking

- Recursive Chunking
- Document Specific Chunking
- Semantic Chunking
- Agentic Chunking

Why Semantic Chunking?

Semantic chunking offers the following advantages:

- Fast
- Efficient
- Accurate

Therefore, we chose semantic chunking over the other methods.

4.1.3 Similarity Search

Similarity Search is the way of RAG to get the targeted related part of the pdfs or chunks to embed the information that is related to the current input in the prompt and get more accurate informative results

4.1.4 embedding in the prompt

And after we retrived the chuncks we embed it in the prompt to be passed to the tokenizer and then to the model after that to get the best accurate result to get the knowledge of the therapist embedded in the tuned model to be your friend therapist

4.1.5 Introduction

To enhance our understanding of emotions in text, we decided to create a classification model that takes a user's input message and predicts the corresponding emotion. This model can be particularly useful in sentiment analysis and mental health monitoring. By accurately detecting emotions, we can improve user interactions and provide more empathetic responses.

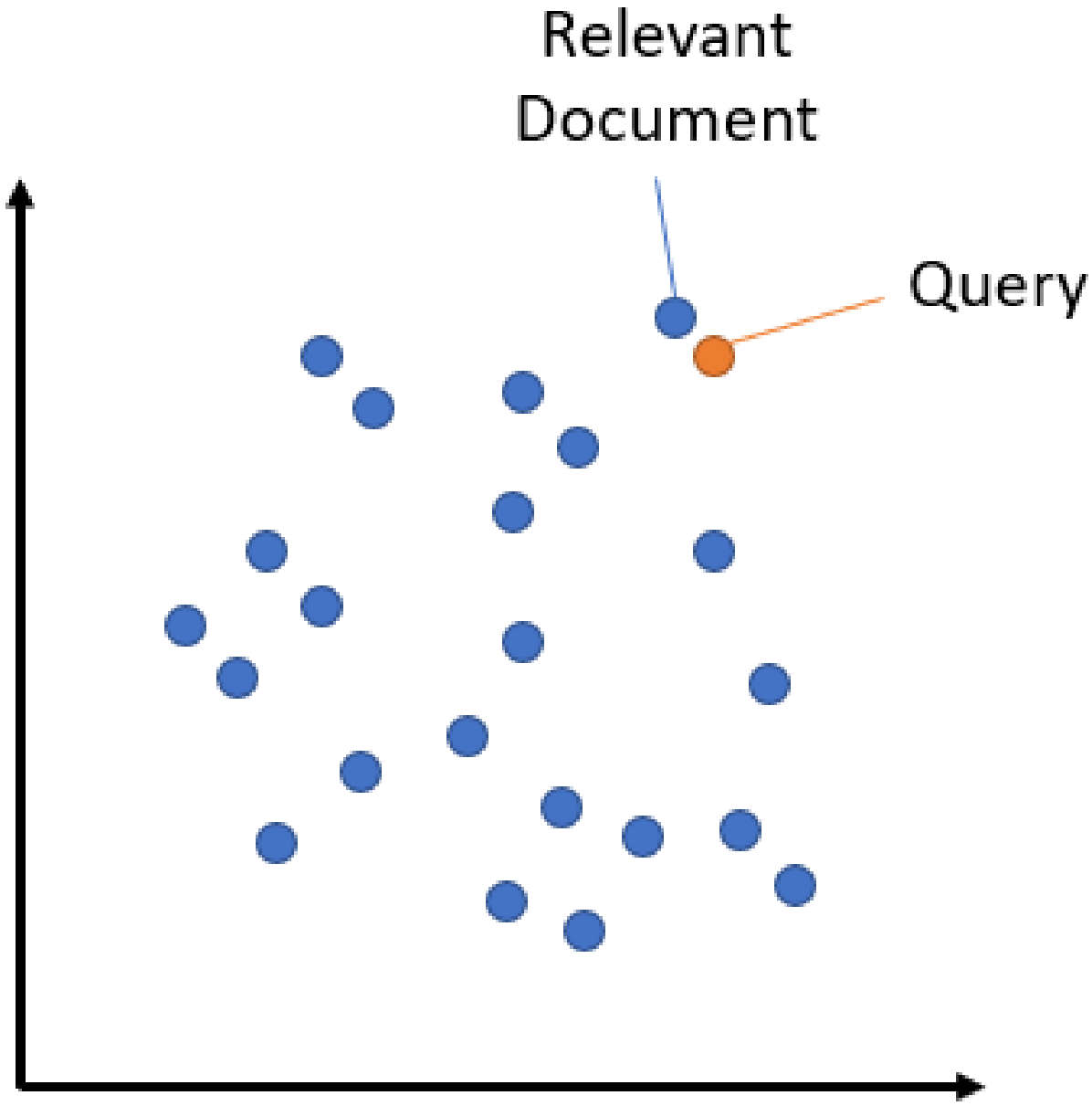


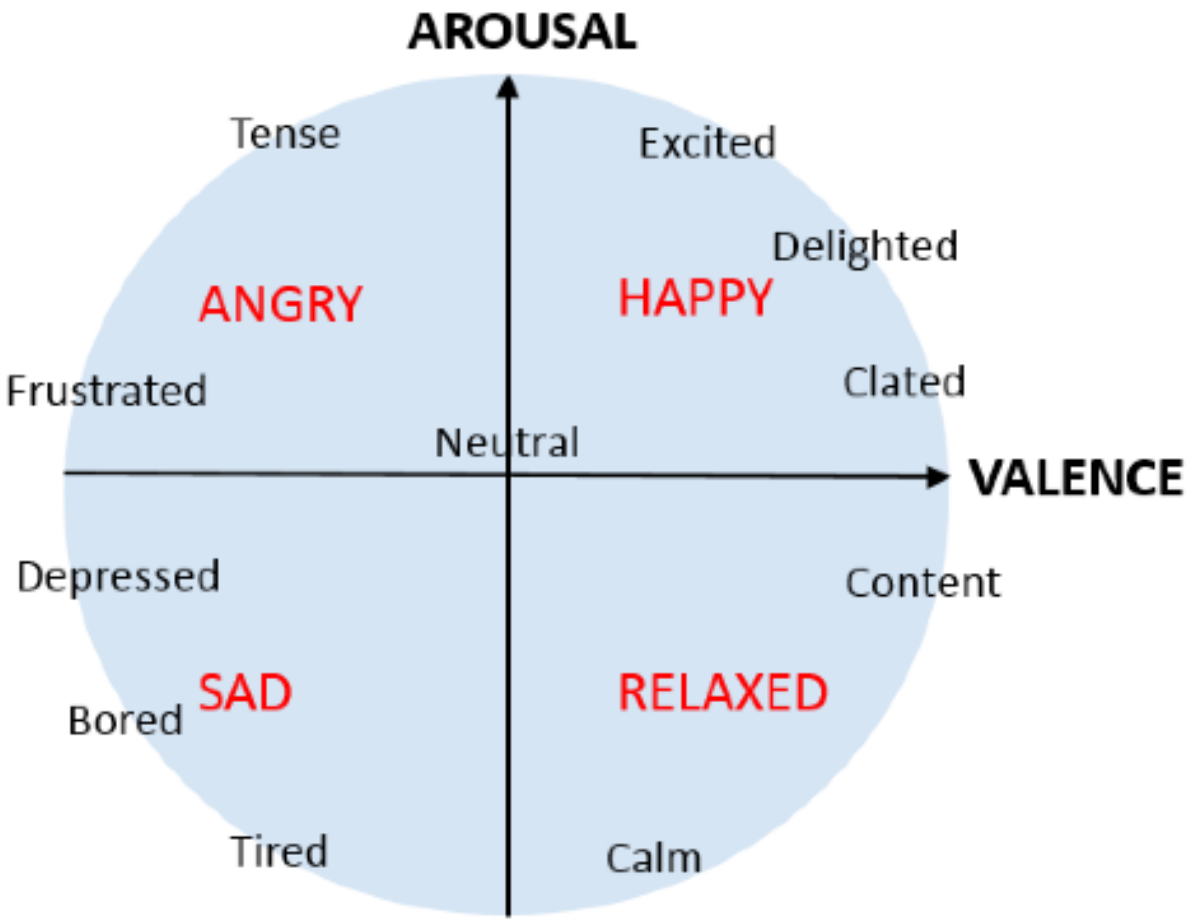
Figure 4.2: showing the chunking process and adding the chunks to the query

4.1.6 Selecting Dataset

We chose an Arabic dataset named arabic-empathetic-conversations for emotion classification. This dataset consists of conversations where each conversation is labeled with an emotion. The dataset has 36,628 rows and 30 unique emotions, organized into three columns: emotion, *context*, and *response*. This rich dataset provides a comprehensive basis for training our emotion detection model.

4.1.7 Preprocessing

To prepare the dataset for training, we performed several preprocessing steps:

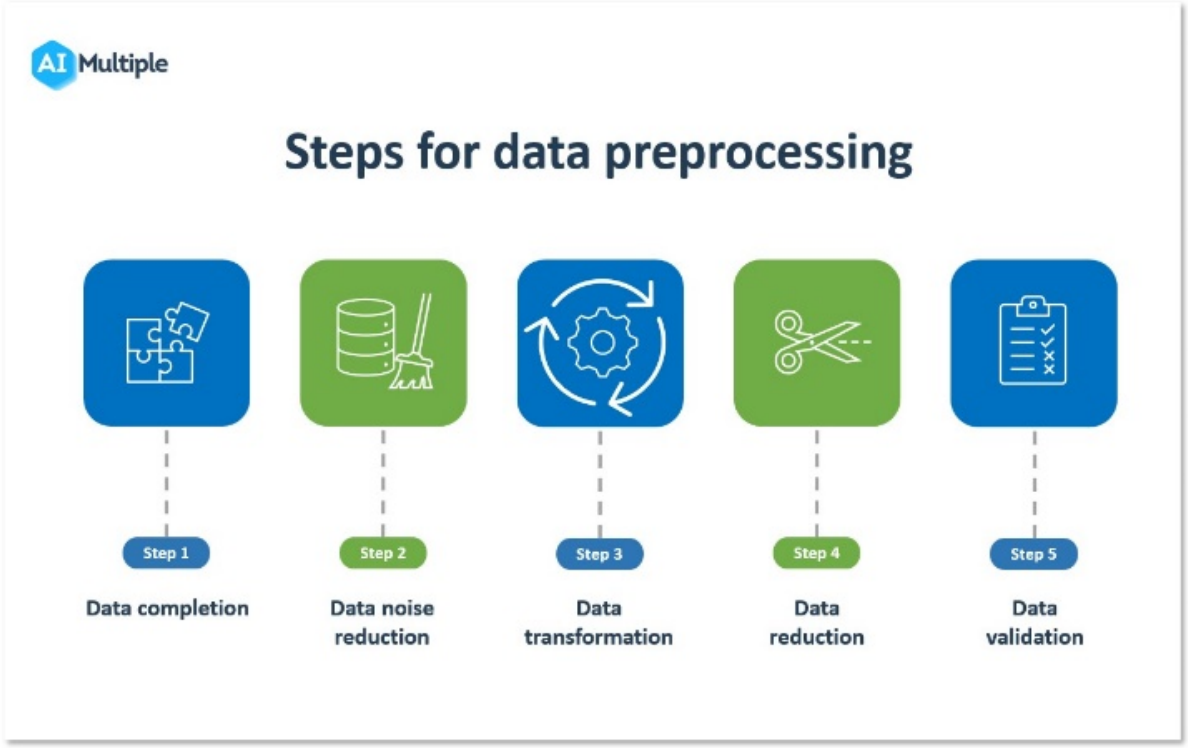


1. **Emotion Selection**: We selected a subset of unique emotions, reducing the dataset size to 5,820 rows. This step helps to focus on the most relevant emotions and reduces the complexity of the model.

2. **Combining Text**: We concatenated the context and response columns into a single column named *text* to represent the entire conversation. This ensures that the model has all the necessary information in a single input field.

3. **Label Encoding**: We converted the emotion labels into numerical indices using a mapping dictionary (lbl2idx). For example, the mapping looked like this: {'joyful': 0, 'sad': 1, 'lonely': 2, 'anxious': 3, 'content': 4}. This numerical encoding is essential for training machine learning models.

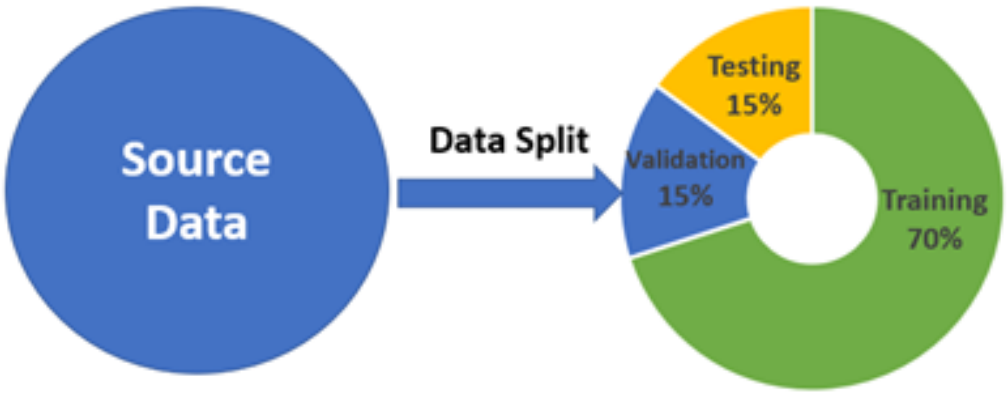
4. **Imbalancing**: To address class imbalance, we used the oversample_data function. This function oversamples the minority classes by randomly duplicating samples until each class has the same number of samples as the majority class. This helps prevent the model from being biased towards the majority class during training. The function utilizes the



resample function from the *sklearn.utils* module.

4.2 Data Splitting

We split the dataset into training, validation, and test sets as follows:



1. **Initial Split**: We divided the dataset into training (70%) and temporary data (30%) to ensure that the model has enough data to learn from while reserving a portion for testing.

2. **Second Split**: We split the temporary data into equal parts for testing and validation (50% each) to fine-tune the model and evaluate its performance.

The final split sizes were:

- Training: 4,074 rows
- Validation: 873 rows
- Testing: 873 rows

This splitting strategy ensures that the model is tested on unseen data, providing a realistic measure of its performance.

4.3 Selecting Model

We chose the BERT model for our classification task, specifically the aubmindlab/bert-base-arabertv02-twitter model. BERT (Bidirectional Encoder Representations from Transformers) models are known for their effectiveness in various NLP tasks, including text classification. BERT's ability to understand context from both directions in a sentence makes it particularly powerful for understanding nuanced emotions.

4.4 Fine-Tuning

1. **Model Loading**: We used the AutoModelForSequenceClassification class from Huggingface, which helps in loading pre-trained models suitable for sequence classification tasks. We specified the number of labels in our dataset during this step. This class automatically adds a classification head to the model, making it ready for our task.
2. **Tokenization**: We used the BERT tokenizer to process the text data. This involved trimming the text to a certain length and adding padding where necessary to ensure uniform input sizes. We also created a custom dataset class to store the tokenized texts and their labels, streamlining the data loading process for training.

4.5 Training

During the training phase, we fine-tuned the BERT model on our preprocessed dataset.

1. **Training Arguments**: We defined various training arguments. These arguments specify where to save the results, how many times to go through the training data, and the number of samples to use in each batch during training and evaluation. They also determine how to adjust the learning rate at the beginning and the strength of weight decay to avoid overfitting. Additionally, the arguments define how often to log and evaluate the model, which in this case is once every epoch.

2. **Trainer Class**: We specified the model to be trained in our trainer class and provided the training arguments defined earlier. It also includes the datasets for training and evaluation and a function to calculate performance metrics. This configuration ensures that the model is trained and evaluated with the specified settings, allowing for a structured and efficient training process.

4.6 Result

After fine-tuning the model on our dataset, we evaluated its performance using the validation and test sets. The results demonstrated the model's ability to accurately classify the emotions associated with the given text inputs. Key performance metrics such as accuracy, precision, recall, and F1-score were used to assess the model's effectiveness.

By following these steps, we successfully developed an emotion classification model that can predict emotions from Arabic text inputs. This model can be further refined and applied to various practical applications to understand and respond to human emotions more effectively. Future work could include expanding the dataset, incorporating additional features, and exploring other advanced models to further enhance performance.

4.7 LoRA and Q-LoRA

4.8 LoRA

LoRA (Low-Rank Adaptation) is a technique to fine-tune large language models efficiently by injecting trainable low-rank matrices into each layer of the transformer architecture. This approach reduces the number of parameters that need to be updated during training, making it computationally efficient and cost-effective, especially for tasks requiring adaptation to specific

domains or languages.

4.9 Q-LoRA

QLoRA (Quantized Low-Rank Adaptation) builds on LoRA by incorporating quantization techniques. Quantization involves representing model parameters with fewer bits, reducing memory and computational requirements. This saves memory and makes the model run faster, especially on devices with less power.

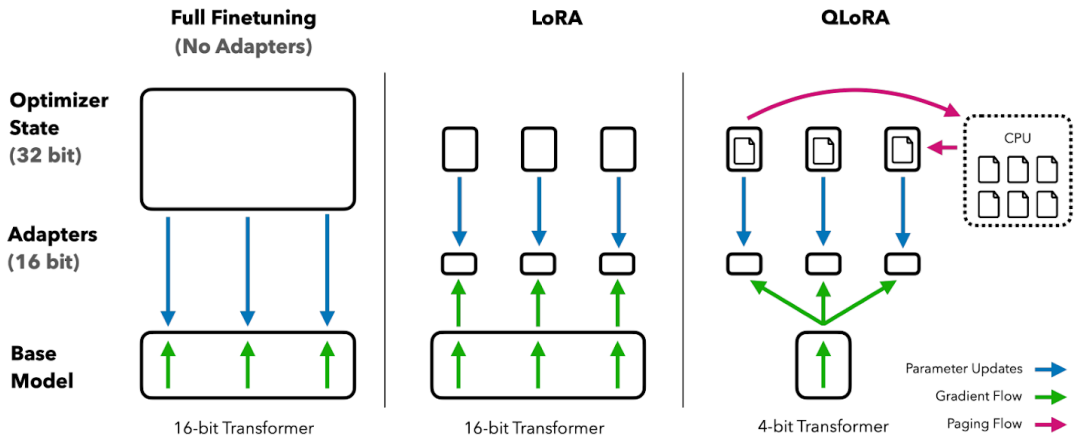


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

4.10 GPU Usage

4.11 Without Q-LoRA

When we train our model on a GPU without using QLoRA, we use the full precision of the model's parameters.

1. **Full Precision Training**: The model uses 16-bit or 32-bit floating point numbers to represent its parameters. This provides high accuracy but requires more memory and computational power.

2. **Memory Usage**: Full precision training uses a lot of GPU memory. For large models, this can quickly fill up the available memory, limiting the batch size and the number of layers we can train simultaneously.

3. **Speed**: Training is fast compared to a CPU, but the high memory usage can sometimes slow down the training process if the GPU memory becomes a bottleneck.

4. **Results**: Full precision training can lead to slightly better results because there's no loss of detail in the model's parameters. However, the difference might not be significant enough to justify the extra resource use.

4.12 With Q-LoRA

When we use QLoRA (Quantized Low-Rank Adaptation) on a GPU, we optimize the training process by reducing the precision of the model's parameters.

1. **Quantization**: We use 4-bit quantization for the model's parameters. This means we represent each parameter with fewer bits, drastically reducing the memory footprint.

2. **Memory Usage**: With 4-bit quantization, the memory usage drops significantly. This allows us to train larger models or use larger batch sizes without running into memory limitations.

3. **Speed**: Training becomes faster because the model is smaller and requires fewer computations. The GPU can process more data in parallel, leading to quicker training times.

4. **Results**: While there's a slight reduction in precision, QLoRA is designed to maintain high performance. The trade-off between memory efficiency and model accuracy is minimal, and in many cases, the results are comparable to full precision training.

4.13 Simple Example

Training a large language model on a GPU without QLoRA might take 10 hours and use up 90% of the GPU's memory, while training the same model on a GPU with QLoRA might take only 6 hours and use just 50% of the GPU's memory.

4.14 Training Process

When we train the model, we follow these steps:

1. **Set Up Quantization**: We configure the model to use 4-bit quantization. This reduces

the amount of memory the model uses.

2. **Prepare the Model**: We prepare the model for training with LoRA. This involves setting up specific parts of the model to be updated.

3. **Define LoRA Config**: We define the settings for LoRA, like how many new parameters to add and where to add them in the model.

4.15 Detail on Each Step

1. **Quantization Configuration**: We use a configuration to tell the model to use 4-bit numbers. This helps make the model smaller and faster.

2. **Prepare for k-bit Training**: We make the model ready for training with fewer bits. This step involves setting up the model's layers.

3. **LoRA Configuration**: We set parameters like $r = 128$ and lora_alpha=32. These numbers control how much change we allow in the model. We also specify which parts of the model to update (q_proj, k_proj, v_proj, o_proj).

4.16 Dataset

4.17 Introduction

We collected our dataset with the help of GPT (AI engine). The dataset consists of raw text conversations on various topics between therapists and their patients. Our goal was to make the dataset as diverse as possible to capture a wide range of interactions.

4.18 Emotion Classification Model For Emotion Detection

4.18.1 Preprocessing

To make the dataset useful for our model, we went through several preprocessing steps:

4.18.2 Text Separation

Each conversation was separated by line. Then splits the conversations using this line.

4.18.3 Detailed Processing

We initialized an empty list called final to store processed chat data and a string history to track the conversation history. We looped through each row in the DataFrame:

- Added the role and message content to the history.
- Identified the position in the chat:
  - **First Chat**: Marked as the beginning if it's the first row and the role is "doctor" (doctor).
  - **Last Chat**: Marked as the end for the last row:
    - For doctors, recorded the conversation with the previous context.
    - For patients, marked it with "end" (closure).
  - **Middle Chats**: Checked if the conversation ID changed:
    - Marked as the beginning if a new conversation started.
    - Marked as the end if a conversation ended.
    - Recorded patient messages with the previous context for regular chats.

4.18.4 New DataFrame Creation

Converted the final list into a new DataFrame dff with columns: *['history', 'patient', 'doctor']*.

4.18.5 Saving and Loading the Dataset

We saved our dataset as a CSV file. Loaded it from the datasets library on Hugging Face. Used train_test_split to split the data into training and validation sets with a test size of 0.2.

4.19 Why Choosing GPT?

Choosing the right model is critical for any machine learning project. For a conversational AI like a therapy chatbot, it's important to select a model that understands and generates natural language proficiently and can be fine-tuned to specific conversation contexts.

4.20 Model Selection

We chose to use GPT (Generative Pre-trained Transformer) for our therapy chatbot because of its strong capabilities in understanding and generating natural language. In our model, we decided to choose AceGPT due to its capabilities in understanding and generating Arabic text, which is essential for our application targeting Egyptian Arabic conversations between therapists and patients.

4.20.1 First Try: AceGPT7B

Firstly, we were confused between choosing acegpt7B and acegpt13B, but because we found the differences in sizes between them were small, we decided to choose acegpt13B due to its big training which is on 13 Billion parameters so it's better to understand our task. In the first try in training this model, it gave us little good answers but it still couldn't chat so well.

4.20.2 Second Try: AceGPT13B-Chat

When using acegpt13Bchat, we noted that the model is too huge and trained on big conversations datasets, so it's not affected with our tuning by Egyptian friend therapist conversations. So we decided to increase the dataset.

4.21 Final Model Choice

Despite initial challenges with limited data and epochs, integrating LoRA and QLoRA has optimized the model's training process. We successfully enhanced the performance of our model to understand and generate Arabic text effectively.