AI-Powered Chatbot For Mental Health

MUHAMMAD EATISAM SADIQ
01-135221-032

Supervisor: Syed Junaid Hussain

Bachelor of Science in Information Technology

Department of Computer Science
Bahria University, Islamabad

December 2025

Certificate

We accept the work contained in the report titled "AI-Powered Chatbot For Mental Health", written by Mr. Muhammad Eatisam Sadiq as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Information Technology.

Approved by . . . :

Supervisor: Syed Junaid Hussain

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr. Kabeer Ahmed Bhatti

_____

Head of the Department: Dr. Khurram Ehsan

_____

Abstract

Mental health is one of the critical problems globally that is impeded by concerns of stigma, price, and accessibility. Although AI-driven chatbots are a promising solution, most current platforms are based on cloud-based architecture due to which the data privacy and latency issue are of a significant concern. In this project, I present a secure and responsive mental health chatbot, Axon Ally, focused on the privacy of the user and providing real-time assistance.

The system is developed on a hybrid architecture, which is a unique mixture of a locally-hosted, fine-tuned Large Language Model (LLaMA 3.2 1B Instruct) and the secure cloud capabilities of Firebase by Google. The frontend is a cross-platform mobile app created in Flutter, which offers a user-friendly and relaxing user interface. As a middleware, the backend is a FastAPI server that receives user requests and serves as the interface with the local AI model and safely performs data transactions with Firebase. Firebase Authentication and Firestore perform user authentication and conversation storage respectively, which is encrypted and provides the security measures of any sensitive data with the industry-standard.

The present project can prove that it is possible to implement a lightweight but efficient AI chatbot that does not rely on third-party AI services. The localization of the language model ensures privacy of the users and reduces the response time. The study confirms that this architecture offers a curious, reliable, and approachable platform to provide early support of mental health to critical disavowal between technological creativity and ethical client treatment. Axon Ally is an example of AI that can be used in sensitive areas as a demonstration of privacy-focused applications.

Acknowledgments

The development of this Final Year Project was tough yet rewarding, and it could not have been accomplished without the help and support of some important people. I would like to take this chance to show my appreciation.

To begin with, I show my utmost and most genuine gratitude to my project supervisor, Syed Junaid Hussain. This project was founded on his mentorship. His feedback and priceless advice were essential from the first conceptualization process to the ultimate implementation. He continuously pushed me to be more critical, to perfect my architecture, and to ensure my work was both technical and ethical. His experience and unswerving encouragement kept me driven and on track, particularly during moments of difficulty. I am grateful for his patience and for giving me the time to help this project achieve its full potential.

I also express my warmest gratitude to the faculty of the Department of Computer Science at Bahria University, Islamabad. The theoretical knowledge and technical skills required for this project were the result of the academic background my professors provided. My research and development work also depended on the resources and infrastructure the department offered. I am grateful for the quality of teaching and the positive learning atmosphere that allows students to consider innovative projects such as this one.

I owe much to my family, who have been my absolute support throughout. Their trust in my capability, and the patience they showed during my many hours of coding, provided the emotional base required to overcome all challenges. Without their unconditional love and faith, navigating the late nights and moments of stress would not have been possible. This project has been driven by the silent but powerful engine of their support.

Lastly, I thank my friends and fellow students. The friendship, new visions, and readiness to hear my thoughts helped me cope with tough moments. I was motivated by the collaborative spirit of my classmates; their brainstorming and words of encouragement were a greater influence than they realize. Every single line of code in this project is an ode to the support that I was lucky to get from all these people. This would never have been finished without them.

MUHAMMAD EATISAM SADIQ
Bahria University
E-8, Islamabad, Pakistan

December 2025

Declaration of Non-Clinical Scope

Statement of Project Scope and Purpose

Axon Ally is planned and executed as a sub-clinical, supportive tool for daily psychological wellness. It is **not a medical device** and is not intended to diagnose, treat, or prevent any mental illness. The system focuses on compassionate listening, companionship, and wellness guidance within established ethical boundaries.

Validation by Mental Health Professionals

To ensure an ethical and responsible design, qualified mental health professionals validated the application through:

1. **Direct Use:** Personal interaction to assess conversational tone and empathy.
2. **Data Review:** Evaluation of anonymized sample conversations and model behavior.
3. **Safety Assessment:** Verification of crisis protocols for high-risk situations.

The reviewers affirm that the project aligns with its role as a non-clinical supportive companion. The system avoids clinical diagnoses or prescriptive advice, operating strictly within the ethical limitations of supportive AI tools.

Reviewed and Validated by:

Fatima Tariq
M.S. Clinical Psychology,
Foundation University, Rawalpindi

Rimsha Altaf
M.S. Clinical Psychology,
NUST, Islamabad

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
UI	User Interface
LLM	Large Language Model
NLP	Natural Language Processing
JWT	JSON Web Token
AES	Advanced Encryption Standard
GDPR	General Data Protection Regulation
SDK	Software Development Kit
HTTP	Hypertext Transfer Protocol
CBT	Cognitive Behavioral Therapy

Chapter 1

Introduction

1.1 Background and Motivation

Mental health has become one of the most important aspects of the entire human welfare in the modern global environment, and it is significantly underserved. There is a massive population that encounters massive obstacles to accessing mental health assistance in a timely and effective manner. These barriers are complex, including the widely practiced social stigma that undermines the goal of seeking help, the costly nature of professional therapy, and the limited number of qualified mental professionals due to the system. This accumulation of forces results in a silent crisis in which many people find themselves single-handedly and in most cases until the situation gets worse.

The quick development of the sphere of Artificial Intelligence (AI), and especially the Natural Language Processing (NLP) is a revolutionary chance to fill this gap of accessibility. Chatbots as conversational AI can be used as a tier one support. These online companions can be made available 24/7, entirely anonymous, and also non-judgmental in nature, thus breaking some of the most important barriers to mental health care entry. Healthy coping skills and a safe emotional outlet can be empowered by offering an immediate, empathetic, and guided conversation to help users become equipped to manage their emotions effectively, as well as have a safe place to express their emotions. The potential has inspired this project titled Axon Ally. It is focused on creating the next generation chatbot that will utilize the power of modern AI as well as be fundamentally designed with the principles of user privacy and data security that cannot be compromised. The scope of user interactions supported by this vision is illustrated in the Use Case Diagram in **Figure 3.1**.

1.2 Problem Statement

The availability of digital mental health solutions has confirmed the need to have them, and the existing technological environment is full of trade-offs that undermine their functionality and reliability. The most common architectural paradigm of the existing mental health chat bots is based on the reliance on large proprietary cloud-hosted AI models, including those offered by high-tech businesses. This is a popular method, which carries with it a complex of severe and frequently unresolved issues:

- **Critical Privacy and Security Risks:** Mental health chatbot is associated with the sharing of highly personal and sensitive data. Users face a great risk of privacy when such conversations are relayed to third-party servers where they are processed. Such risks are the possible data breach, unauthorized access by internal or external parties, and the likelihood of the use of the data in business without any direct user permission. This is the key privacy issue that introduces a significant obstacle to trust and does not allow users to be absolutely transparent and sincere.
- **Performance and Latency Issues:** The interactive type is essential in real-time as it makes a conversation real and more engaging. External reliance on APIs also creates a delay in the network, which may cause apparent delays between a user and a chatbot. Such delays do not only interrupt the flow of conversation, but also reduce the presence experience, and may result in the users giving up on the application.
- **Deficiencies in Empathetic Understanding:** Most of the current systems especially those that are not designed on advanced models resort to simple, rule based logic or to generic, pre-written responses. This leads to robotic, impersonal, and unfamiliar interactions with contextuality. To an emotionally distressing user, an inappropriate or insensitive reaction is more disorienting than supportive.

According to this study, there is an evident and urgent necessity to find an alternative solution the one that can provide intelligent, sophisticated, and sensitive conversations without making its users to even lose their right to privacy, which is their basic right. The main goal of Axon Ally is to fill this very gap through the development of a strong and powerful chatbot that will be powered by "privacy-by-design" principle.

1.3 Project Objectives

This project is a presentation of the design, implementation, and evaluation of Axon Ally, a new mental health chatbot constructed on a new technology stack chosen thoughtfully and based on a modern technology stack. Its components are a cross-platform mobile application developed using Flutter, an efficient back-end written with FastAPI, and a

secure authentication system and database with Google Firebase. Axon Ally is driven by a locally hosted LLaMA 3.2 1B Instruct model to provide conversational intelligence. The reason why this model was selected was its balance between performance and efficiency and has been carefully fine-tuned on an enormous scale specialized corpus of mental health related talks of more than 1.7 million.

Having the AI model execute on local infrastructure and linking it to the frontend through a secure tunnel, the system will not make the sensitive user data processed by other, third-party AI services. This project can thus be outlined as having the following objectives:

- To design and establish a full-fledged mobile application that delivers a supportive, context-aware and low-latency chat experience.
- To design a safe and scalable user authentication system with the help of Firebase Authentication and make sure that user accounts are secured and regulated based on the industry best practices.
- To enable and develop a persistent data storage with Firebase Firestore, in which all records on user conversations are encrypted on the client and then stored, with the records offering an end-to-end security assurance.
- In a bid to justify the architectural decision of a hybrid local-cloud system, demonstrating that one can ensure the complete privacy of the users during core AI interactions and use the cloud to do non-sensitive managerial duties.

1.4 Project Scope

The scope of this project is carefully defined to ensure the delivery of a high-quality proof-of-concept within the technical and temporal constraints of a final-year project.

1.4.1 Included Features

This project deliverables entail:

- A complete mobile app that is cross-platform and developed using Flutter with a complete flow of user experience, including Splash, login, chat, and profile screens.
- A specific backend server developed with FastAPI that will be the secure middleware and coordinates the communication between the firebase services, the mobile client, and the local LLaMA model.
- A real-time bidirectional chat interface with the capability of persisting the message history, meaning that people are able to look back and reflect on the conversations they had previously.

1.4.2 Excluded Features & Limitations

To ensure that the project scope is narrow and feasible, it is explicitly stated that the following are not included in the scope or rather serve as limitations:

- **Model Reasoning Capabilities:** The chosen LLaMA 3.2 1B model with its high efficiency in its size has poorer capabilities of complex reasoning and long-term memory than significantly larger, state-of-the-art models (e.g. GPT-4).
- **Development Environment Constraints:** The use of ngrok tunnel is only a development and demonstration phase solution. To have a production ready deployment, it would be required to migrate the backend and model to a dedicated, secure server with a permanent public IP address and domain.
- **Ethical Boundary:** This is one of the basic principles of this project in the sense that Axon Ally is not a medical device, but a supporting tool. The chatbot is clearly programmed to avoid making clinical diagnoses, treatment or substituting the crucial need of a competent human therapist.

By establishing these functional boundaries and ethical limitations, the project ensures that the development focus remains strictly on user privacy and technical reliability. These constraints act as a roadmap for the subsequent chapters, which detail the transition from these high-level objectives into a robust technical architecture, beginning with a survey of the current technological landscape in mental health AI. This structured approach allows for a rigorous evaluation of the system's performance and its viability as a privacy-focused mental health companion.

Chapter 2

Literature Review

Axon Ally development is premised on the extensive literature and technology review on three main grounds: digital mental health apps, the history of conversational AI, and the essential area of data privacy and security. This review discusses the state-of-the-art issue in order to determine the considerable advances that have been achieved and the ongoing issues that persist to date hence putting the research gap addressed by this project in perspective [4].

2.1 The Landscape of Digital Mental Health Interventions

This section examines the current ecosystem of digital mental health tools and how they have evolved to meet increasing global demand for accessible care.

2.1.1 Prominent Commercial Chatbots

Chatbots like **Woebot** [1], **Wysa** [2], and **Youper** [3] have been the first to utilize chatbots in mental wellness. They are frequently developed based on existing therapeutic models, and mostly Cognitive Behavioral Therapy (CBT). They take users on guided exercises, mood tracking, and psychoeducation proving the possibility of conversational agents to provide evidence-based interventions. The effectiveness of this method is proven by studies, including the one that was carried out on Woebot and demonstrated statistically significant results of improved symptoms of depression and anxiety among users [1].

Nonetheless, when critical analysis is done on these systems, a similar architectural dependency is found out. Most of them are based on potent, proprietary, cloud-based Large Language Models (LLMs), including the GPT series offered by OpenAI to drive their chat capabilities [5]. The two key issues that this dependency brings are first, it requires a reliable and uninterrupted internet connection which may be a limitation in the underserved

areas. Secondly, and more importantly, it also casts deep concerns over the privacy and security of the highly sensitive user data that is being cranked on third-party servers [6].

2.2 Advancements in Conversational AI and Language Models

The intelligence of modern chatbots is directly proportional to the rapid evolution of transformer-based architectures and massive datasets.

2.2.1 The Rise of Open-Source LLMs

Although proprietary models such as GPT have taken the news, the open-source community has come up with a force of substitutes. Models like LLaMA series by Meta [7], Falcon by TII [8] and Mistral models by Mistral AI [9] have proven to be highly competitive to their closed-source equivalents. The presence of such models has opened access to state-of-the-art AI, allowing scientists and developers to create their own solutions, and not be tied to the ecosystem of one vendor.

2.2.2 The Critical Role of Fine-Tuning

Studies have always shown that the performance of general-purpose LLMs out-of-the-box does not meet the performance requirements of specialized and high-stakes fields such as mental health. Fine-tuning the process of training a pre-trained model on a smaller dataset that is domain specific is necessary [10]. Research has revealed that refining a model on a collection of therapeutic dialogues, supportive online forum discussions and psychological texts has a huge impact on the capacity of the model to produce empathetic, contextually attentive, and secure responses [11]. This procedure converts the model behavior of the model to the realities and ethical demands of mental health assistance, decreasing the propensity of this model to produce generic and unhelpful or even harmful content. The Axon Ally project is based on this idea, and the I m LLaMA model is fine-tuned on a large mental health-specific corpus [7].

2.3 Data Privacy and Security in AI Applications

Perhaps the biggest barrier to the mainstreaming and acceptance of AI in mental healthcare is data privacy. This section explores the vulnerabilities inherent in current architectures and the regulatory standards required to protect user information [15].

2.3.1 Architectural Vulnerabilities

Single points of failure are not desired by malicious attacks; centralized and cloud based storage architectures are convenient, yet they form single points of failure, which make them convenient. The repercussions of data breach of such a system might be disastrous to users [13]. Contemporary privacy-saving methods are highly needed to eliminate these threats. These are strong end-to-end encryption whereby the data on the client is encrypted and only the user can decrypt it, and anonymization of data, whereby personally identifiable information is removed in the data used to generate analytics or model enhancements.

2.3.2 The Viability of Local and Edge Computing

With the recent advancements in technology, it has become more and more possible to execute powerful, refined language models directly on local or sometimes even on the edge hardware, like a personal computer or even a high-end mobile phone [14]. This can be frequently obtained by methods such as model quantization that decrease the accuracy of the weights of the model thus making it smaller and less computationally intensive with negligible performance cost [12]. The method provides unlimited control over the data pipeline. It fully avoids transmitting sensitive data to third parties servers to have it processed hence reducing a significant commercial category of threats to privacy [6]. It also enables it to set up tailored safety filters and logging systems which lie entirely within the authority of the developer.

2.4 Identifying the Research Gap and Contribution

The given literature review indicates that there is a major conflict within the existing digital mental health environment. This gap provides the primary motivation for the current project.

2.4.1 Distinction from General-Purpose Language Models

The key point to notice in this research gap is that there is a difference between specialized chatbots such as Axon Ally and general-purpose types such as ChatGPT or other AI helpers. Although general-purpose models are simply too powerful, they are not geared to the context of high-stakes mental health support [11]. The value proposition of Axon Ally is characterized by its benefits in three main domains, namely privacy, specialization, and user experience.

- **Privacy by Architecture:** The General-purpose AIs are nearly always cloud-based, i.e. user conversations are sent, and stored, and commonly used by the provider to train a model. The hybrid architecture of Axon Ally has a local AI model that gives

an architectural assurance that sensitive conversational data is not processed by a third party [14].

- **Specialization and Safety:** General-purpose models are trained to be know-it-alls of immense internet information. They do not have the guardrails and understanding tones needed in mental health discussions. The model of the Axon Ally, on the contrary, is an expert. Its practice with a specific mental health dataset aligns its behavior to promulgate supportive, empathetic, and its behavior to be within a safety framework that encompasses crisis intervention measures [11].
- **Focus and User Experience:** General-purpose AI is distracting to use. Axon Ally offers an environment that is specifically designed. The user interface is also meant to be relaxing and simple with all the emphasis of the user being on the supportive talk only. It is an exclusive, risk-averse environment, and not a versatile device.

A detailed comparative analysis of these differences is summarized in **Table 2.1**, which contrasts the functionality and privacy measures of general models versus the specialized Axon Ally system.

Table 2.1: Comparison: Axon Ally vs. General-Purpose LLMs

Feature	**General-Purpose LLMs (e.g., ChatGPT)**	**Axon Ally**
Core Purpose	Broad knowledge, multi-tasking. A generalist.	Empathetic listening and well-ness guidance. A specialist.
Model Training	Trained on massive, diverse in-ternet text.	Fine-tuned on a large, specific mental health dataset.
Data Privacy	Processed and stored on third-party cloud servers.	Processed locally; no third-party AI sees the chat.
Safety	Relies on general safety filters.	Implements specific protocols for crisis intervention.
User Experience	Multi-purpose interface, can be distracting.	Dedicated, calming interface designed for focus.

Axon Ally will directly fill this gap. It makes contributions to the field by suggesting and developing a new hybrid architecture. This architecture is a strategic integration of a high-performance, lightweight, locally-hosted LLM of all core conversational processing with the safe and scalable database-as-a-service (DBaaS) and authorization of Google Firebase [16]. This solution is a solid, user friendly and highly responsive solution that does not sacrifice on user privacy, and is directly responsive to the fundamental challenges of usability, security, and trust within the next generation of digital mental health solutions.

Chapter 3

Requirement Specifications

The blue print towards a successful software project is a comprehensive requirement definition. The functional, non-functional, user, and system requirements of Axon Ally are carefully defined in this chapter. The specifications are used as the ultimate guide during the design, development, and testing, in order to achieve the project with the core objective that satisfies the requirements of the end-users of the product.

3.1 Existing System

Current digital mental health interventions, such as commercial chatbots (e.g., Woebot, Wysa), predominantly rely on cloud-based architectures. While effective, these systems transmit sensitive user data to third-party servers for processing, raising significant privacy concerns. Furthermore, they require constant internet connectivity to function, leading to latency issues and lack of accessibility in offline scenarios.

3.2 Proposed System

The proposed system, Axon Ally, overcomes these limitations by introducing a privacy-first, hybrid architecture. By hosting a fine-tuned Large Language Model (LLaMA 3.2 1B) locally, the system ensures that sensitive conversational data is processed on the user's device or a secure local server, minimizing data exposure. The specific requirements to achieve this are detailed below.

3.3 Requirement Specifications

This section details the technical specifications categorized by their functionality and performance constraints to ensure a robust system.

3.3.1 Functional Requirements

Functional requirements determine the particular behavior and functionality of the system. They state what the system has to do. In the case of Axon Ally, user management, conversational interaction, and data persistence are the core requirements.

1. **Secure User Account Management System**
    - **FR1.1 (Registration):** The system will also have an easy to use interface where new users can subscribe to an account with a valid email address and a secure password. The strength of passwords will be enforced.
    - **FR1.2 (Login):** The system will enable the registered users to log in safely. After a successful authentication, the system will create and administer a session token to be used to authorize future requests.
    - **FR1.3 (Logout):** The system will support a way of users to log out safely, which will nullify their active session token.
    - **Technology:** The functionality will be applied through Firebase Authentication service.
2. **Core Application Interface**
    - **FR2.1 (Screen Navigation):** The Flutter application will be designed using four main, navigable screens, including a Splash Screen that will be used to initialize the application, a Login/Registration Screen to allow users to access it, a main Chat Screen where the user will be able to communicate with others, as well as a Profile Screen that will contain information specific to the user.
3. **Persistent and Secure Chat History**
    - **FR3.1 (Message Storage):** All pairs of messages (user input and AI response) of a chat session will be stored in Firebase Firestore database securely.
    - **FR3.2 (Data Association):** The individual conversation logs stored will be explicitly associated to unique User ID (UID) of the authenticated user, thus data separation and privacy.
    - **FR3.3 (Data Retrieval):** The system will have the capability to acquire the entire conversation history of a user that is logged in to the system so that it can be shown inside the application.
4. **Real-Time Conversational Engine**

- **FR4.1 (Interaction):** The Chat Screen will enable users to type text messages and show the AI-generated responses in close real-time, which will provide a dynamic and interactive flow of conversation.
- **FR4.2 (Backend Processing):** To enable this real-time interaction, the FastAPI backend will be used and will handle user input by sending it to the AI model and then the response.

5. **User Profile and Activity Dashboard**
   - **FR5.1 (Information Display):** The Profile Screen will show the account details of the logged-in user (e.g. email address).
   - **FR5.2 (Chat Analytics):** The profile will calculate and show basic analytics, like the number of messages sent and received in total, to motivate the users to use it.
   - **FR5.3 (History Access):** The profile screen will be the entrance by which the users will be able to access and read their past conversations.
6. **AI Model Middleware**
   - **FR6.1 (Backend Service):** The FastApi backend will constitute a special purpose middleware server that will isolate the frontend client of the complexity of the AI model.
   - **FR6.2 (Model Communication):** The backend will be in charge of all communication with the locally hosted LLaMA model through in-house HTTP requests.
7. **Context-Aware Conversation**
   - **FR7.1 (Session Context):** The backend system will keep a short-term conversational context of each active user session. This will enable the chatbot to refer to some of the recent components of the conversation to give better fitting and relevant responses.

A prioritized summary of these functional requirements is provided in **Table 3.1**.

Table 3.1: Summary of Important Functional Requirements by Priority

Ref ID	Requirement Summary	Priority
FR1	**Authentication:** User registration and secure login via Firebase.	High
FR3	**Data Privacy:** Encryption and isolation of chat logs per user.	High
FR4	**AI Interaction:** Real-time message processing and inference.	High
FR6	**Middleware:** Secure communication between Client and AI Model.	High
FR3.3	**Persistence:** Retrieval of previous conversation history.	Medium
FR7	**Context Awareness:** Maintaining short-term session memory.	Medium
FR5	**Analytics:** Displaying user statistics and message counts.	Low

3.3.2 Non-Functional Requirements

The quality attributes of the system are set by non-functional requirements. The three most critical requirements for Axon Ally are detailed below, along with the method used to verify them.

NFR1 (Usability): The application's user interface must be clean, intuitive, and calming, minimizing cognitive load on potentially distressed users. All primary functions should be accessible within three taps.
Verification: This was tested during User Acceptance Testing (UAT). Participants were given tasks (e.g., "Find your past chats") and the number of taps and time taken were recorded. A System Usability Scale (SUS) survey was also conducted.

NFR2 (Security): All information being exchanged between the client (Flutter app), the backend (FastAPI), and the database (Firebase) should be encrypted during transmission through the industry-standard HTTPS (TLS 1.2 or more).
Verification: Verified by analyzing network traffic using proxy tools to ensure no plain-text data was visible. Database security rules were tested by attempting to access a user's data using a different user's authentication token (which correctly resulted in "Permission Denied").

NFR3 (Performance): Normal network and server load conditions should not exceed 2.5 seconds in terms of the average end-to-end response time between the receipt of a message by the user and the appearance of the response on the screen.

Verification: Performance testing was conducted by logging timestamps at the moment the "Send" button was pressed and when the response rendered. An average was calculated over 50 test interactions.

3.3.3 System Requirements

This section outlines the specific hardware and software environments required for the development and execution of Axon Ally.

3.3.3.1 Hardware Requirements (Development)

- **Development Machine:** Personal Computer having a modern multi-core processor (e.g. Intel Core i7 or AMD Ryzen 7).
- **Memory:** At Least 16GB of RAM to support both the operating system and the development tools and the AI model at the same time.
- **GPU:** The efficient local inference of the quantized LLaMA model requires an NVIDIA GPU that has CUDA support and at least 8 GB of VRAM (e.g., RTX 3070 and above).

3.3.3.2 Software Requirements

- **Frontend:** Flutter SDK (version 3.24 or more), Android Studio or Visual Studio Code.
- **Backend:** Python (3.10 or above), FastAPI, Uvicorn, ngrok.
- **AI:** Transformers library PyTorch, a quantized version of the LLaMA 3.2 1B model.
- **Database:** A Google firebase project is configured with firestore and authentication.

3.4 Use Cases

The functional requirements of the system can be mapped to specific Use Cases that describe the interaction between the User (Actor) and Axon Ally (System). These mappings are detailed in **Table 3.2**.

Table 3.2: Detailed Use Case Descriptions

ID	Actor	Description	Map FR
UC-01	New User	The user provides a valid email and password to create a new profile in the system.	FR1.1
UC-02	Registered User	The user authenticates with credentials or Google Sign-In to access the application features.	FR1.2
UC-03	Auth User	The user types a message in the chat interface. The system processes the input and returns an empathetic AI response.	FR4.1
UC-04	Auth User	The user navigates to their profile or history section to view past conversations stored securely in the database.	FR3.3
UC-05	Auth User	The user views their account details and usage analytics (e.g., total messages exchanged).	FR5.1
UC-06	Auth User	The user ends the current session, ensuring the security token is invalidated on the device.	FR1.3

3.4.1 Use Case Diagram

The diagram below, shown in **Figure 3.1**, visually represents the scope of the system and the interactions described in the use cases above.

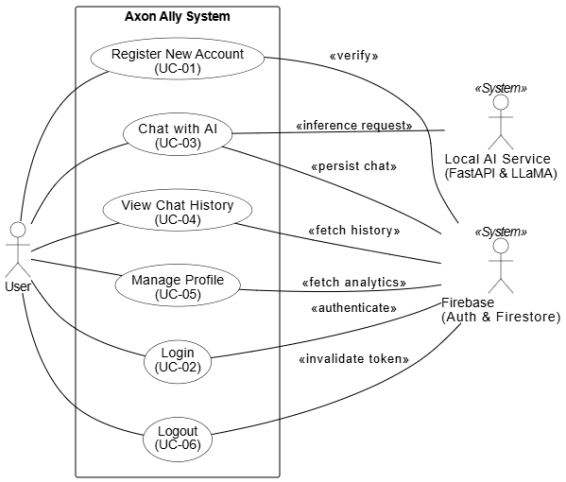

Figure 3.1: Use Case Diagram covering all defined Use Cases

3.5 User Stories

The requirements are then presented in user stories, which are user-centric requirements of the end-user so that the development process is user-centric.

US1: As a novice user who cares about the privacy details, I would like to have a secure account that only requires my email address and a password so that I could have access to the chatbot and feel safe that all of my conversations are stored privately and are associated with me alone.

US2: As an experienced user, I would like to be in the position to view and scroll through all my past chats easily in order to be in a position to reflect my thoughts, emotional state and what advice I got.

US3: When accessing it as an active user, I would like to be shown some simple statistics, such as the number of chats that I have had in my profile page since it will provide me with a feeling of advancement and activity using the application over time.

US4: As the developer creating and testing the application, I would like to use and test the AI model using a local FastAPI server to guarantee the utmost data privacy in the development stage and facilitate quick and high-speed changes without the need to have a complicated deployment pipeline.

Chapter 4

Design

After the full specification of requirements, this chapter describes the design of the Axon Ally system at the architectural level as well as component-level level. Design phase converts the requirements that are already identified into a clear technical outline that will be used in the implementation. The major philosophy behind this design is to build a system, which is modular, secure, scalable as well as performant and highly focusing on the user privacy.

4.1 System Architecture

The architecture of Axon Alley was developed in such a way that it builds around the principles of modularity and separation of concerns. Multi-tier (or n-tier) architectural pattern was chosen strategically in order to separate the key functions of the system in a logical way.

4.1.1 The Three-Tier Architectural Pattern

This pattern ensures that the presentation logic, application logic, and data management are decoupled, allowing for easier maintenance and testing. The specific components of this architecture are illustrated in **Figure 4.1**.

- **Presentation Layer (Frontend):** This is the front-end aspect of the application which renders the interface and records the user interactions. It was a strategic decision to leverage Flutter.
- **Application Layer (Backend Middleware):** This layer serves as the brains of the operation making it process all the business logic. It is a simple middleware that is developed using FastAPI.

- **Data Layer (Backend-as-a-Service):** This layer performs all the data storage and user identity using Google Firebase.

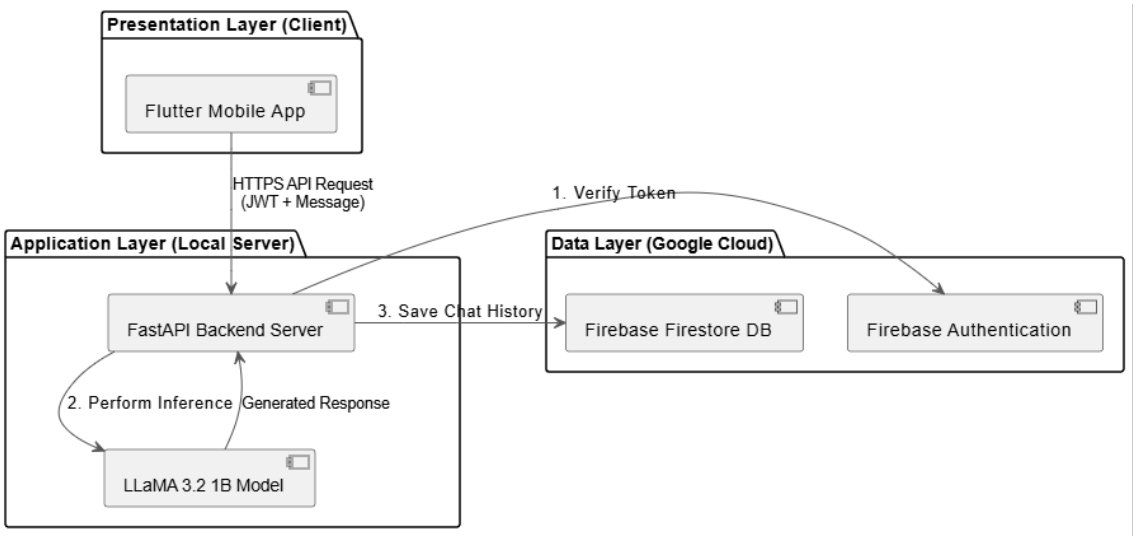

Figure 4.1: Axon Ally System Architecture

4.2 GUI Design

This section provides the detailed design of the system inputs and outputs relative to the user interface. The visual language is designed to be calming and intuitive for users in distress.

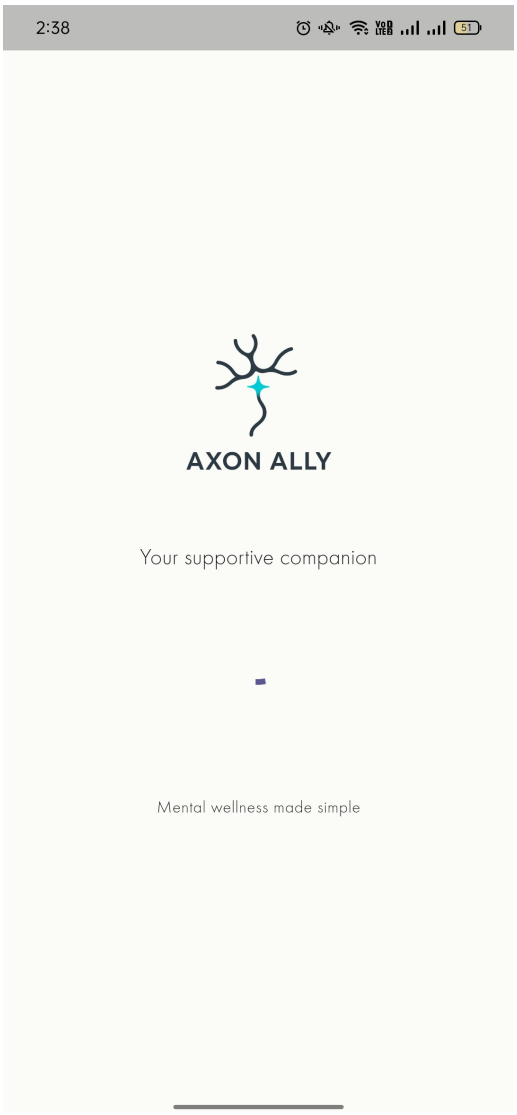

Figure 4.2: The Splash Screen shown on application startup.

The splash screen serves as the initial touchpoint for the user. It features a minimalist aesthetic with the Axon Ally neural icon, designed to establish a sense of technical reliability and professional calm immediately upon application launch while the system initializes background services.

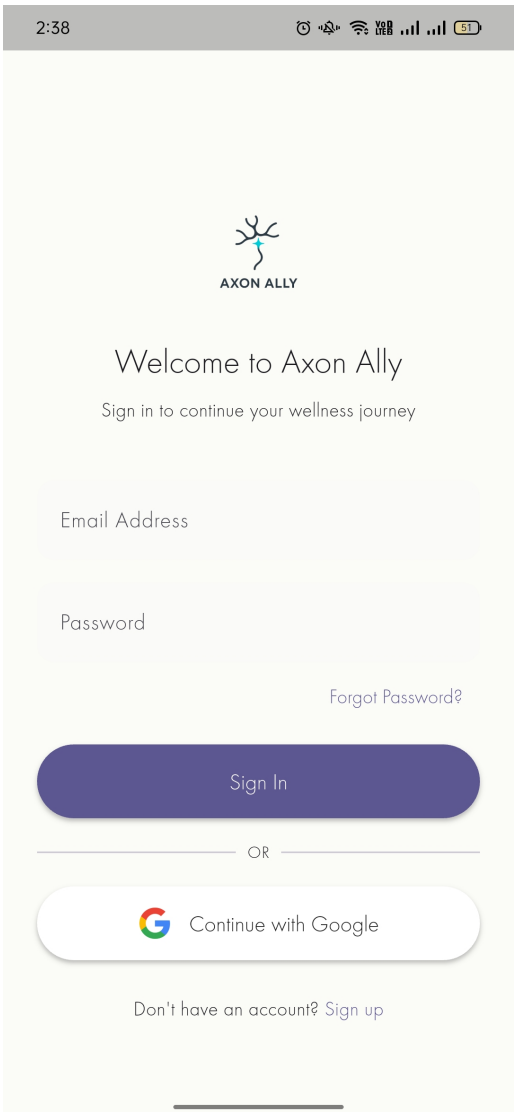

Figure 4.3: The main Login Screen with options for Email/Password and Google Sign-in.

The login interface is focused on accessibility and low friction. By offering both traditional email-based entry and a one-tap Google Sign-in option, the system ensures that users can access their supportive environment quickly without the burden of navigating complex authentication forms.

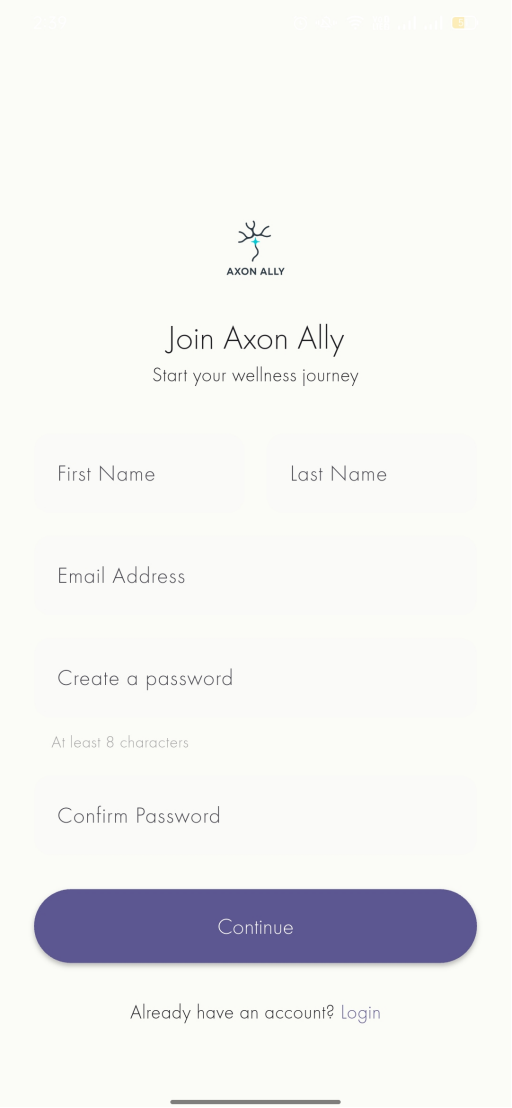

Figure 4.4: The User Registration (Sign Up) Screen.

During the registration process, the interface emphasizes data integrity and account security. The form includes real-time validation for email formats and enforces strict password complexity rules, ensuring that every user starts their journey with a well-protected account.

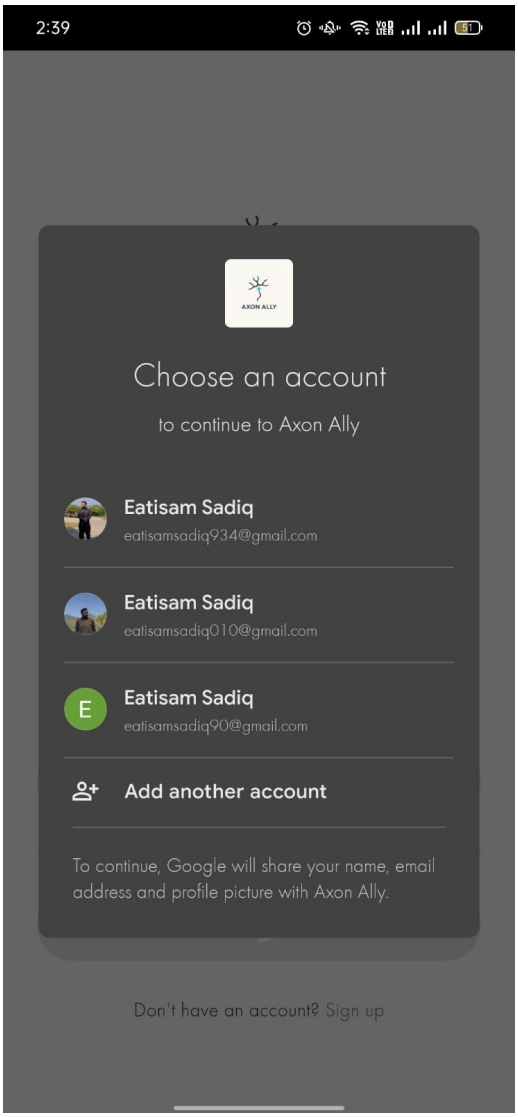

Figure 4.5: The Google Account selection interface for Google Sign-in.

Integration with industry-standard OAuth 2.0 via Google provides an extra layer of trust. As seen in **Figure 4.5**, this standard interface allows users to leverage existing secure accounts, removing the need to remember new credentials while maintaining a high level of security.

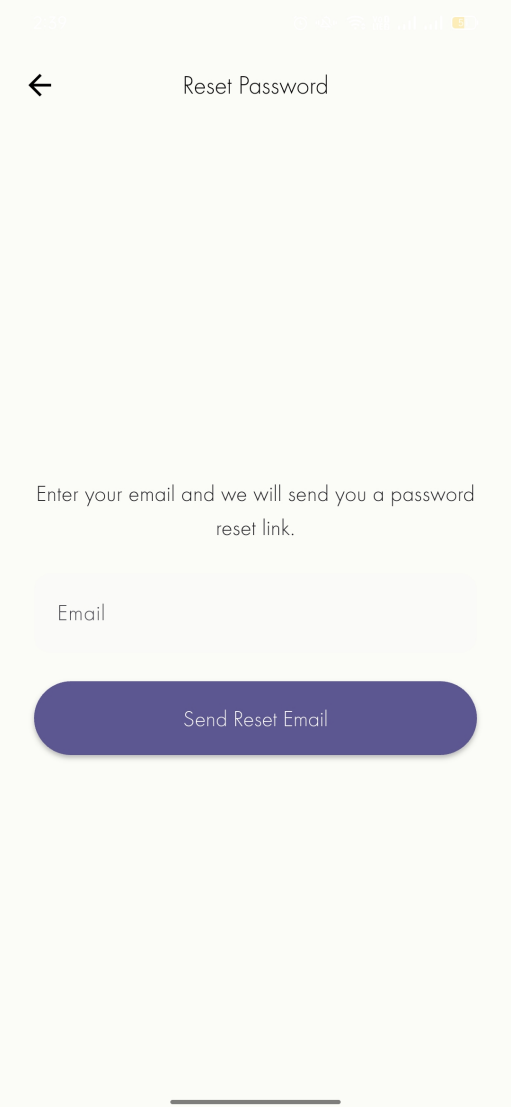

Figure 4.6: The interface for resetting a forgotten password.

To prevent user frustration and loss of access, a streamlined password recovery interface is included. This ensures that users can regain access to their private conversation history securely via email-based verification links, maintaining the continuity of their mental health support journey.

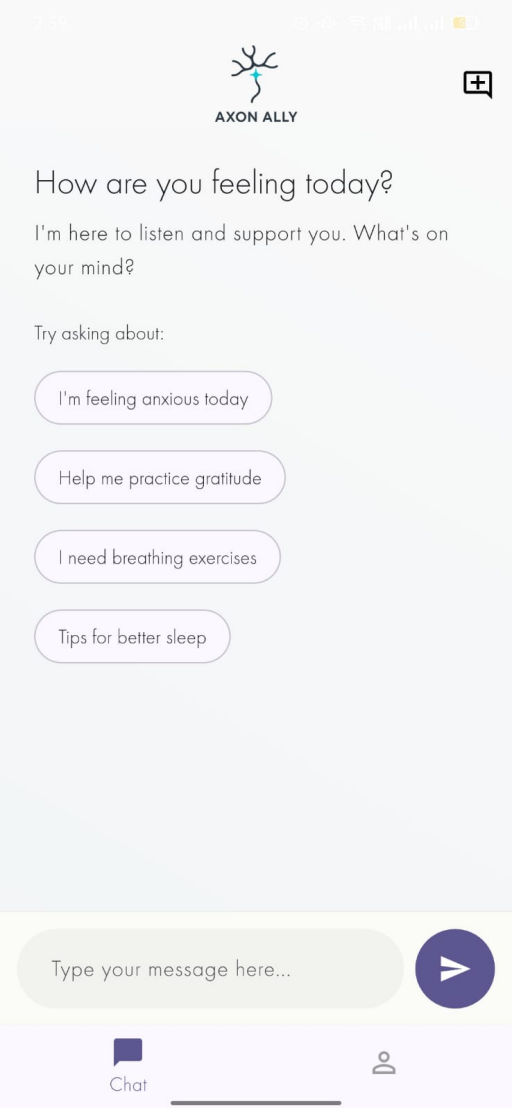

Figure 4.7: The core Chat Screen with prompt suggestions.

The chat screen, visualized in **Figure 4.7**, is the primary functional area of the application. It includes clickable prompt suggestions to help users who may find it difficult to start a conversation, fostering an interactive and supportive dialogue through a clean, distraction-free messaging interface.

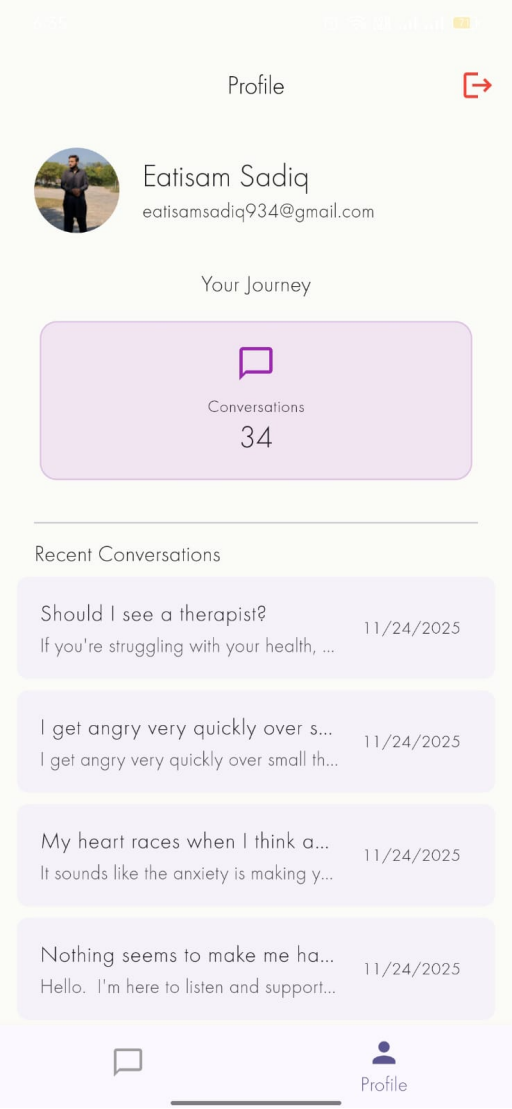

Figure 4.8: The User Profile Screen showing conversation count and recent chats.

The profile screen acts as a dashboard for self-reflection. By displaying analytics such as total conversation counts and a list of recent interactions, it encourages users to track their progress over time and provides a central hub for managing account-specific settings.

These interface components are built with responsiveness at their core, utilizing Flutter's Flexbox-based layout engine to ensure that the visual experience remains consistent across various mobile screen dimensions. This design philosophy minimizes the cognitive load on potentially distressed users, providing a stable and reliable platform for mental health assistance.

4.3 Design Constraints

The system design deals with specific constraints related to privacy and hardware. These factors dictate the choice of models and deployment environments.

- **Local Hardware Resources:** The local hosting of the LLaMA model requires a GPU with at least 8GB VRAM (e.g., NVIDIA RTX 3070). This limits the server-side deployment to machines with sufficient graphical processing power.
- **Network Dependency:** While inference is local, the authentication and database services rely on Firebase, requiring an active internet connection.
- **Privacy Compliance:** All architectural decisions are constrained by the need to ensure no unencrypted user chat data leaves the local/secure environment.

4.4 Design Methodology

The project follows an iterative design methodology. The design phase converts the requirements identified in Chapter 3 into a clear technical outline. We utilize standard Object-Oriented analysis and Unified Modeling Language (UML) diagrams to model the system's behavior and structure before implementation.

4.5 High Level Design

To further clarify the system's operation, Data Flow Diagrams (DFD) are used to visualize how data moves through the Axon Ally system from the user to the local model.

4.5.1 DFD Level 0 (Context Diagram)

The Context Diagram represents the top-level overview of the system interactions, as shown in **Figure 4.9**.

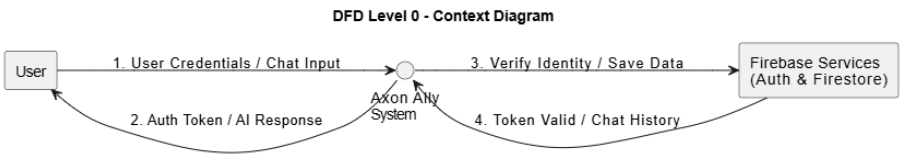

Figure 4.9: DFD Level 0 - Context Diagram

4.5.2 DFD Level 1 (Detailed Flow)

The Level 1 DFD, depicted in **Figure 4.10**, decomposes the main system process into three distinct sub-processes to show detailed data routing and logic.

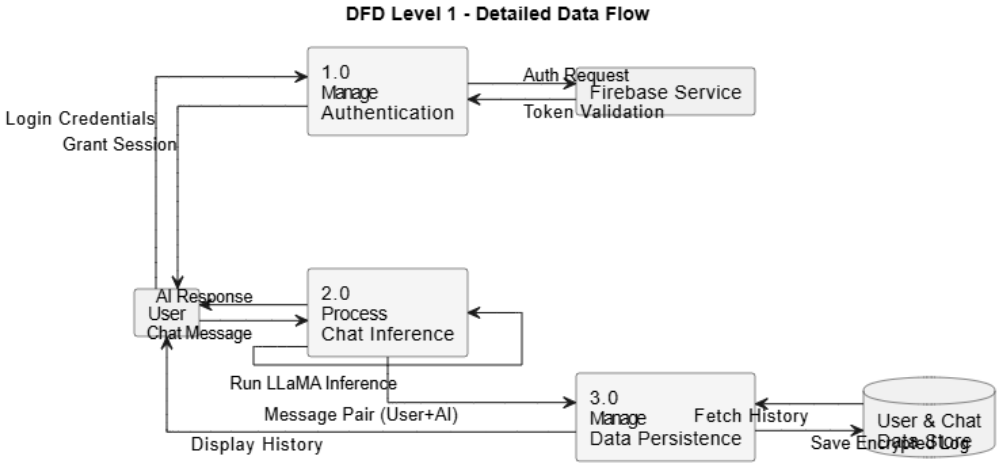

Figure 4.10: DFD Level 1 - Detailed Data Flow

4.6 Low Level Design

This section provides detailed design descriptions using Activity and Sequence diagrams to model the exact logic and timing of system operations during a chat session.

4.6.1 Activity Diagram (Process Logic)

The logical flow of a single chat transaction, including authentication checks and safety filtering, is mapped out in **Figure 4.11**.

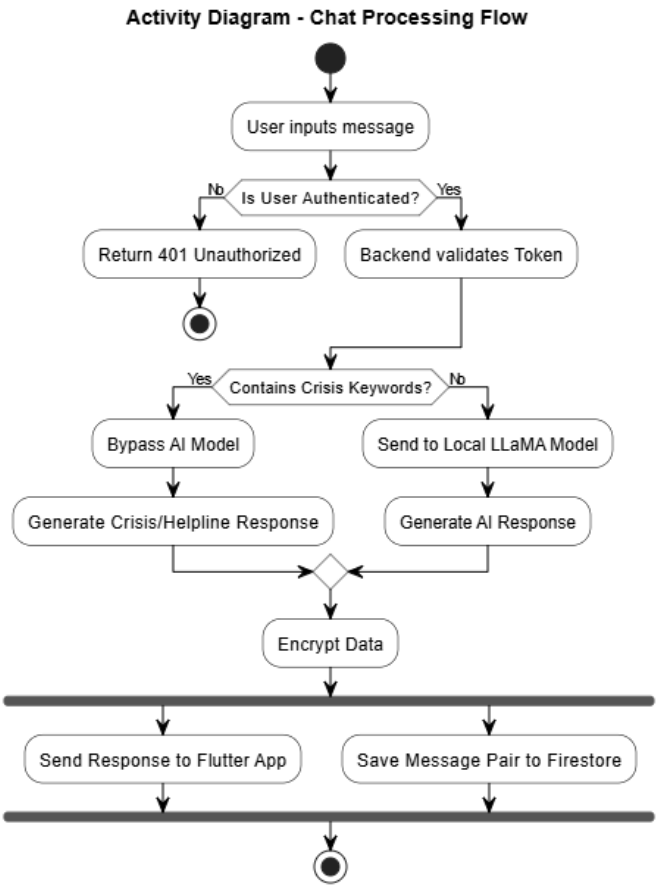

Figure 4.11: Activity Diagram of Chat Processing

4.6.2 Sequence Diagram (Timing & Interaction)

The interaction between the user, the mobile client, the FastAPI middleware, and the AI model is illustrated in the Sequence Diagram in **Figure 4.12**.

1. **Authentication:** Firebase Authentication verifies the user token.
2. **Message Transmission:** Flutter sends HTTPS request to FastAPI.
3. **Backend Processing:** FastAPI validates token and conveys message to LLaMA.
4. **AI Inference:** LLaMA model produces a text response.
5. **Response and Persistence:** Response is sent to client and saved to Firestore.

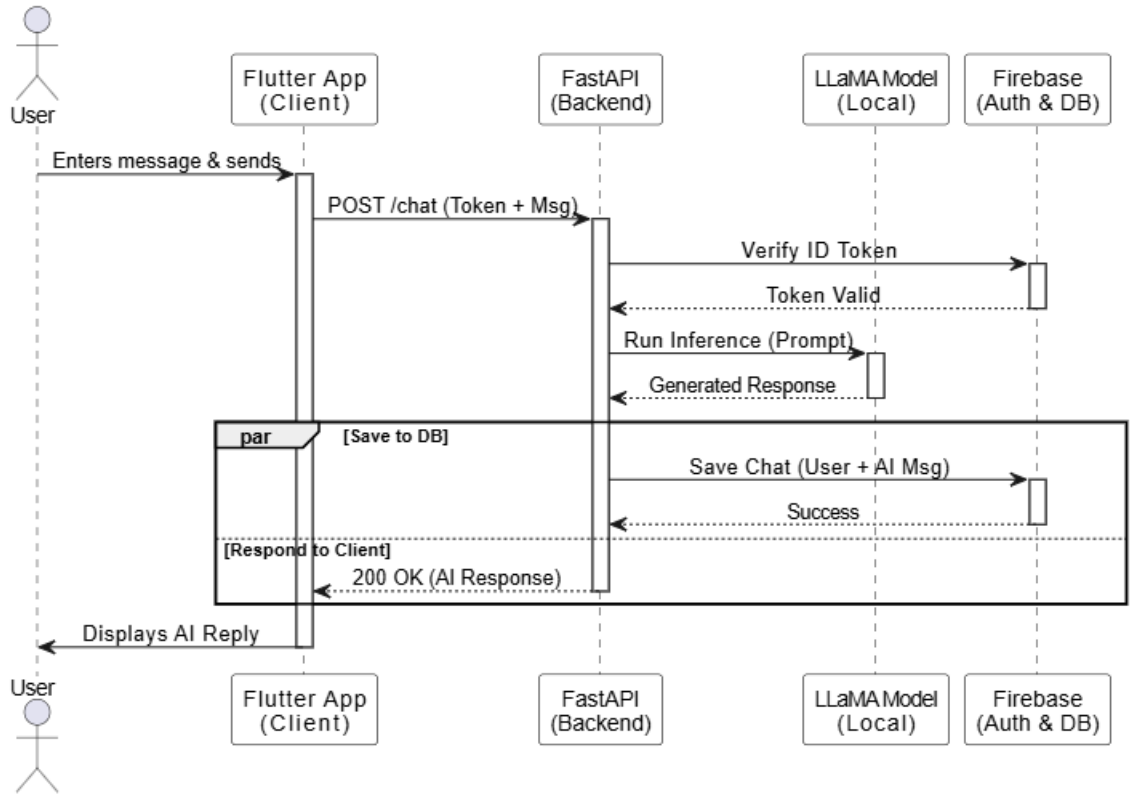

Figure 4.12: Chat Flow Sequence Diagram

4.7 Database Design

The data is designed as NoSQL using Firebase Firestore to offer flexibility and scalability for conversational data structures.

4.7.1 Data Model

The model is optimized for rapid read/write operations of chat messages. The core collections are described below.

- **`users` Collection:** Identified by the unique Firebase UID. Includes `email`, `createdAt`, and `total message count`.
- **`conversations` Collection:** Stores message arrays. Each document contains a `messages` array of map objects (text, sender, timestamp).

4.7.2 Entity Relationship Diagram (ERD)

The relationship between users and their respective encrypted chat histories is documented in **Figure 4.13**.

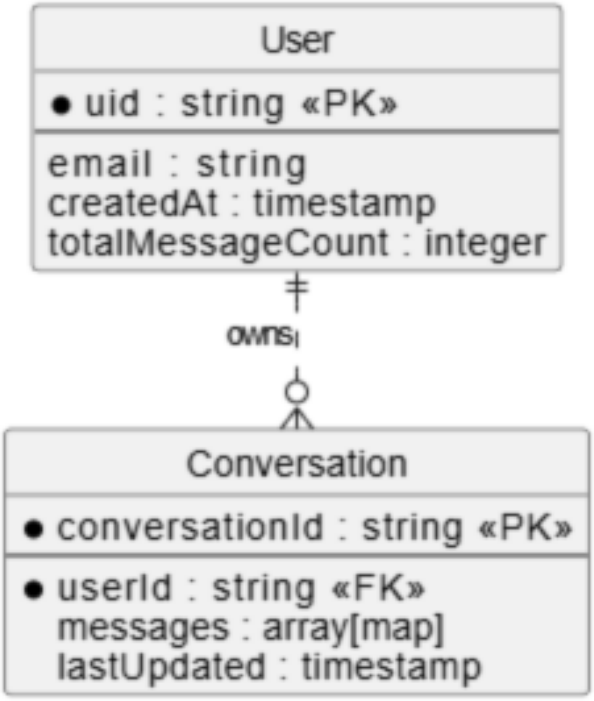

Figure 4.13: Entity Relationship Diagram (ERD)

4.8 Security Architecture

Axon Ally places security as a cornerstone of the design that is non-negotiable. A Multi-layered security strategy was applied to ensure data integrity, as detailed in the following points.

- **Encrypted Communication in Transit:** All messages between the Flutter client and backend are sent via HTTPS to prevent man-in-the-middle attacks.
- **Robust Authentication:** Firebase Authentication handles user identities safely without storing raw passwords on the local backend.
- **Granular Security Rules:** Server-side Firestore rules ensure that users can only read or write to their own specific conversation documents.
- **Encryption at Rest:** Chat logs utilize Firestore's default encryption along with an extra client-side hashing layer for sensitive metadata.
- **Data Privacy by Design:** The AI model is hosted locally, ensuring the raw contents of user discussion never move off the secure environment.

Ultimately, this security architecture is designed to build a "circle of trust" between the user and the local AI model. By ensuring that the most sensitive part of the interaction—the conversation itself—never leaves the secure environment of the host device, the system sets a new standard for ethical AI applications in mental healthcare.

The synthesis of these diverse design decisions—from the high-level tier separation and NoSQL database modeling to the granular logic flow depicted in the activity and sequence diagrams—provides a robust technical blueprint for the implementation phase. By prioritizing modularity and separation of concerns, the architecture ensures that Axon Ally can adapt to evolving user needs while maintaining its core commitment to uncompromising data privacy. This holistic design methodology serves as the critical bridge between abstract user requirements and a functional, secure reality, ensuring that every architectural component works in harmony to deliver a seamless and supportive mental health companion. This structured design approach not only fulfills the immediate performance criteria but also offers a future-proof framework that is ready for the technical complexities detailed in the following chapter on system implementation.

Chapter 5

System Implementation

The chapter describes the actual implementation of the Axon Ally system, which puts the design requirements in the chapter before it into working code. The implementation process was segmented into four main work streams which included: the establishment of the development environment, the development of the frontend application, the development of the middleware backend, and the configuration of the AI model to perform local inference.

5.1 System Architecture (Implementation View)

The technical stack chosen for this project emphasizes a balance between modern cross-platform capabilities and high-performance local AI processing. This section details the specific versions and tools used during the construction phase.

- **Frontend Framework:** Flutter SDK version 3.24 and the Dart SDK of the programming language were installed. Android Studio was the main individual Development Environment (IDE) used for building and debugging the user interface.
- **Backend Framework:** Backend Python 3.10 was used to develop the middleware utilizing the FastAPI framework . The code editor was Visual Studio Code, utilizing Pydantic for data validation.
- **Database and Authentication:** Google Firebase console provided the infrastructure for real-time Firestore database management and secure user Authentication.
- **AI Model and Libraries:** Fine-tuned LLaMA 3.2 1B Instruct model was deployed using Hugging Face `transformers` and `torch` libraries for efficient execution.

5.2 Frontend Implementation (Flutter)

The Flutter application was developed with heavy emphasis on developing a clean, responsive and maintainable codebase, following a modular component-based approach.

5.2.1 State Management and Logic

The implementation of state management is critical for a smooth user experience. The `Provider` package was selected because of its simplicity and effectiveness. The state of the conversation, such as the list of messages and the state of loading when having to wait to receive a response from the AI, was handled with the help of a special class named `ChatProvider`.

5.2.2 Firebase Integration

Integration with cloud services allows for secure data persistence. The official `FlutterFire` suite was integrated with Firebase, specifically including the `firebase_core`, `firebase_auth`, and `cloud_firestore` plugins. A persistent session was established based on the firebase auth stream which automatically verifies the users log in status on opening the app and directs the user to the right screen.

5.3 Backend Implementation (FastAPI)

The back-end was adopted in FastAPI as the effective and secure middleware between the mobile front-end and the heavy AI model.

5.3.1 API Endpoints

The backend logic is exposed via RESTful endpoints designed for performance and security. The desired structure of request and response bodies was defined by Pydantic models. The key functional endpoint is:

- **`POST /chat`**: This endpoint is secured using Bearer Token authentication and acts as the gateway to the LLaMA model inference engine.

5.3.2 Secure Tunneling with ngrok

During the development lifecycle, connectivity between the mobile device and the local server is essential. The locally running Uvicorn server (the FastAPI app running on localhost:8000) was made public over the internet with the help of ngrok. This created a secure public URL over HTTPS that allowed the mobile app to communicate with the local backend across different network environments.

5.4 Model Hosting and Local Inference

The local hosting of the LLM is the most critical implementation milestone, ensuring that user data never leaves the developer-controlled environment.

5.4.1 Model Quantization and Loading

In order to run the Large Language Model on consumer-grade hardware, the LLaMA model was **quantized** to 4-bit precision using the NF4 technique. This significantly minimized the memory footprint and accelerated the inference process while maintaining a high level of conversational quality.

5.4.2 Inference Process

The pipeline converts raw user text into AI-ready prompts. Upon a request reaching the `/chat` endpoint, the backend converts the message typed in by the user to a particular format known to the fine-tuned model as a prompt. This prompt is encoded and inputted to the `generate` method to produce an empathetic response.

5.5 Firebase Configuration and Security

The cloud components of the system were configured with a "security-first" attitude to protect user metadata and account information.

- **Authentication Method:** Configured with Email/Password and Google Sign-in to provide flexible access while maintaining security.
- **Security Rules:** A fundamental rule applied to the Firestore database was `allow read, write: if request.auth.uid = resource.id;`, ensuring that users can only access their own private chat logs.

5.6 Deployment Strategy

The Axon Ally deployment design was a flexible design that could be used to support the speed of the iterative process in the development phase.

5.6.1 Development Deployment Environment

To ensure maximum privacy during testing, the system was deployed in a local-first configuration. This setup, visually summarized in the Development Deployment Diagram in **Figure 5.1**, illustrates how the different services interact within a secure local environment.

- **Local Hosting:** Local deployment of the FastAPI server and a quantized version of LLaMA model was done on an NVIDIA RTX 3070 GPU.
- **Secure Tunneling:** Used to bridge the gap between the mobile hardware and the local inference engine via an encrypted HTTPS tunnel.

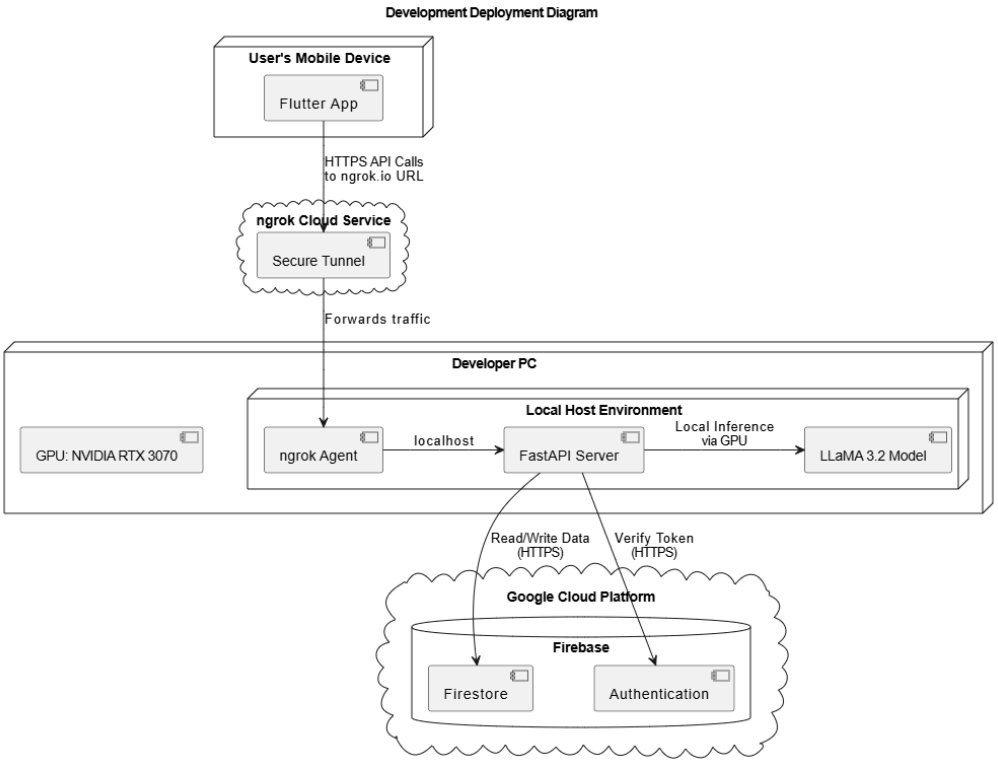

Figure 5.1: Development Deployment Diagram showing the local setup

5.6.2 Proposed Production Deployment Strategy

Moving from a proof-of-concept to a production-ready system requires a more scalable infrastructure.

- **Cloud or On-Premise Hosting:** The FastAPI application will be transitioned to a dedicated cloud server (such as AWS EC2 or DigitalOcean) with GPU support to handle concurrent users.
- **Containerization with Docker:** To guarantee environment uniformity across different servers, the whole backend application will be packaged using **Docker** containers.

5.7 Ethical Framework and User Safety

Technical implementation is accompanied by an ethical framework to ensure the safety of users in emotional distress.

- **Clear Role as a Supportive Tool:** The application includes clear disclaimers informing the user that it is a supportive companion, not a medical substitute.
- **Crisis Intervention Protocol:** The system has an inbuilt safety protocol to detect high-risk keywords associated with self-harm and provide immediate helpline details.

The successful implementation of these components marks the transition of Axon Ally from a theoretical design into a functional tool. By combining the cross-platform flexibility of Flutter with the high-performance inference of quantized Large Language Models, the system demonstrates that privacy-focused AI is not only possible but highly efficient. The modular nature of this implementation allows for future updates, such as model fine-tuning or the addition of new therapeutic features, without disrupting the core architectural integrity. This implementation phase provides the necessary foundation for the rigorous testing and performance evaluation detailed in the following chapter.

Chapter 6

System Testing and Evaluation

Testing and evaluation stage is a significant part of the software development life cycle, which is aimed at confirming that the deployed system addresses the required requirements, and also to confirm its quality in general. This chapter outlines the method of testing, the particular test scenarios, which have been conducted, and the outcomes of performance evaluation and user feedback sessions.

6.1 Testing Approach and Methodology

An extensive testing plan was developed in order to make sure of the reliability and strength of the Axon Ally application. The testing was further broken down into three major types:

- **Unit Testing:** Dedicated to the back-end of the logics of the Fast API application using **PyTest**.
- **Integration Testing:** Aimed at testing the flow of communication and data in the architecture of the various layers.
- **User Acceptance Testing (UAT):** Aimed at testing the system as an end-user with five volunteer users.

6.2 Test Plan and Scenarios

This section details the execution of the formalized test plan, focusing on verifying system stability across both standard and exceptional conditions.

6.2.1 Functional Testing (Use Cases)

Functional testing was conducted to verify that all primary system interactions function as intended for various user roles. The following table, **Table 6.1**, outlines the test cases derived directly from the system Use Cases (UC-01 to UC-06).

Table 6.1: Functional Test Cases based on Use Cases

Test ID	**Use Case**	**Expected Result**	**Status**
TC-01	Register (UC-01)	User creates account with valid email/pass; account appears in Firebase Auth; user redirected to Chat.	Passed
TC-02	Login (UC-02)	Registered user enters valid credentials; valid JWT token is issued; access granted to main app.	Passed
TC-03	Chat (UC-03)	User inputs message; AI responds within threshold; message pair appears in UI instantly.	Passed
TC-04	History (UC-04)	User navigates to profile/history; previous chat logs are fetched from Firestore and displayed correctly.	Passed
TC-05	Profile (UC-05)	Profile screen loads; displays correct email and accurate count of total messages sent.	Passed
TC-06	Logout (UC-06)	User clicks logout; session token invalidates; user redirected to Login screen; back button disabled.	Passed

6.2.2 Advanced Scenarios (Security & Reliability)

In addition to the happy-path use cases, the system was subjected to rigorous edge-case testing to ensure security and stability under stress. These scenarios and their respective outcomes are documented in **Table 6.2**.

Table 6.2: Security and Edge Case Scenarios

Test ID	Scenario & Expected Result	Status
TC-07	**Invalid User Login:** When the user enters an incorrect password or non-existent email, a clear and user-friendly error message is displayed.	Passed
TC-08	**Unauthorized API Access:** An attempt to call the backend `/chat` endpoint via Postman without a valid Firebase ID token results in a `401 Unauthorized` error.	Passed
TC-09	**AI Safety Filter:** When input contains high-risk keywords (e.g., self-harm), the system bypasses the LLM and returns a hard-coded crisis intervention message.	Passed
TC-10	**Network Failure:** If internet connectivity is lost during message transmission, the app catches the exception and displays a "Connection Error" snackbar with a retry option.	Passed

6.3 Performance Evaluation

One of the measures used to determine the quality of the system was performance, especially the latency of the chat interaction. Maintaining a responsive dialogue is essential for user engagement in a mental health context.

- **End-to-End Latency:** The mean round-trip time was **2.3 seconds**. This is well within the acceptable threshold for real-time Large Language Model interactions.
- **AI Inference Time:** Approximately 1.8 seconds on average, demonstrating the efficiency of the 4-bit quantization process.
- **Database Operations:** Fewer than **200 milliseconds**, ensuring that message history and user profiles are loaded almost instantly.

6.4 User Feedback and Qualitative Evaluation

The qualitative feedback acquired during the User Acceptance Testing (UAT) stage gave important information on the user experience. Five volunteers interacted with the system and provided the following feedback.

- **Usability and Interface:** Rated as easy to use, calming, and non-intimidating. Users specifically praised the prompt suggestions for making starting a conversation easier.
- **Response Quality and Empathy:** Responses were described as empathetic, relevant, and noticeably more supportive than typical rule-based chatbots.
- **Trust and Privacy:** Knowledge of local processing significantly increased user trust, with participants feeling more comfortable sharing sensitive thoughts.

The results from these comprehensive testing rounds confirm that Axon Ally successfully fulfills both its functional and non-functional objectives. By passing all critical security tests and maintaining high performance despite the complexity of local AI inference, the system establishes itself as a reliable and ethically sound mental health tool. The feedback from the UAT phase further validates the architectural decision to prioritize privacy, as it directly translated into a higher level of user confidence and therapeutic engagement. This rigorous evaluation phase ensures that the platform is ready for real-world application, as discussed in the concluding chapter.

Chapter 7

Conclusions

This final chapter synthesizes the results of the design, implementation, and testing phases. It reflects on the primary objectives of the Axon Ally project and outlines a roadmap for future technical and clinical enhancements.

7.1 Conclusions

The development of Axon Ally successfully demonstrates that it is possible to create a sophisticated, empathetic mental health companion without compromising user privacy. This project was able to illustrate the overall design, implementation, and assessment of a chatbot built on the foundation of data security and real-time responsiveness. The system is a viable solution to the delivery of private and supportive conversations by creatively combining a locally hosted and fine-tuned LLaMA 3.2 (1B) model with a modern technology stack, including a Flutter front-end, a FastAPI middleware, and Firebase for cloud-based managerial tasks.

The hypothesis of the project was that a hybrid architecture would be effective to take advantage of the scalability of cloud access in non-sensitive operations such as user management and data storage while guaranteeing full confidentiality for highly sensitive conversational data. This hypothesis is proved by the successful implementation and positive evaluation of the system. In general, Axon Ally has managed to reach its main goals:

- **Performance:** An all-purpose, cross-platform chatbot that is able to produce empathetic responses with low-latency (2.3s round-trip time) was demonstrated to be helpful and reliable.

- **Security:** An integrated system of multi-layered security, featuring end-to-end encrypted chat history and local AI inference, has established a new standard for user privacy.
- **Integration:** An end-to-end communication pipeline between the Flutter client, the FastAPI server, and the local LLaMA model was successfully verified through rigorous testing.

The project stands as a powerful demonstration of a proof-of-concept in the digital mental health realm. It proves that privacy is not a feature that must be sacrificed for quality; rather, it can be the core pillar upon which a supportive user experience is built.

7.2 Future Work

While the current implementation of Axon Ally provides a strong foundation, the modular nature of its architecture allows for several promising avenues of future development. These enhancements would aim to increase both the clinical utility and the technical scalability of the platform.

- **Advanced Model Fine-Tuning and Personalization:** The chatbot conversational skills may be enhanced by increasing the sample of the fine-tuning data with more various and subtle examples of therapeutic conversation.
- **Interactive Mood Tracking and Analytics:** An interactive mood tracking feature can be added, where users record their emotional state daily to visualize patterns over time.
- **Expanded Accessibility with Multilingual Support:** To achieve a global audience, the chatbot can be trained on high-quality and multilingual datasets to support users in their native languages.
- **Transition to a Scalable Cloud Deployment:** Beyond the proof-of-concept phase, the backend can be migrated to a scalable cloud platform using Docker and Kubernetes to handle thousands of concurrent users.
- **Bridging the Gap with Professional Care:** A "Therapist Portal" could be added, allowing users to voluntarily and securely share anonymized conversation logs with licensed professionals for deeper clinical insight.

7.3 Final Remarks

The history of developing Axon Ally was a potent quest into the area of the convergence of technology, psychology, and ethics. This project shows that as AI becomes more integrated

into our daily existence, it is our duty as engineers to design systems that are not only intelligent but also humane, respectful, and above all, trustworthy.

Axon Ally is evidence of the fact that technology can and should be a force for good in the field of mental health. Although it is not meant to supersede the priceless nature of human connection and professional therapy, it is a significant move in the right direction toward establishing the availability of data-safe and truly useful tools. It provides a secure environment in which people can share their emotions, learn coping mechanisms, and take the initial courageous step on the path to emotional health.

Ultimately, the success of Axon Ally serves as a reminder that the goal of engineering in healthcare is to empower the individual. By placing the "privacy-by-design" principle at the center of the development lifecycle, this project has successfully bridged the gap between cutting-edge Large Language Model capabilities and the ethical requirements of mental health support, creating a companion that is as secure as it is supportive.

Appendix A

Key Source Code Snippets

A.1 FastAPI Chat Endpoint (main.py)

The implementation of the major `/api/chat` endpoint within the FastAPI backend is represented in the following Python piece of code. This role reads the message of the user, assembles an appropriately formatted prompt, feeds it into the local LLaMA model to be inferred and then provides the generated response.

```
@app.post("/api/chat", response_model=ChatResponse)
async def chat(chat_request: ChatRequest):
    # Removed special unicode checkmark to prevent font errors
    print(" /api/chat endpoint hit! (INSECURE MODE)")

    if not model or not tokenizer:
        raise HTTPException(status_code=503, detail="AI model unavailable.")

    system_prompt = (
        "You are Axon Ally, a world-class compassionate and highly skilled "
        "mental health counselor. Your purpose is to provide professional, "
        "empathetic, and supportive guidance."
    )

    # Use tokenizer.apply_chat_template for robust formatting
    messages = [{"role": "system", "content": system_prompt}]
    for turn in chat_request.history:
        messages.append({"role": turn['role'], "content": turn['content']})

    prompt = tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True
    )

```

```
    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
    terminators = [
        tokenizer.eos_token_id,
        tokenizer.convert_tokens_to_ids("<|eot_id|>")
    ]

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_new_tokens=300,
            eos_token_id=terminators,
            do_sample=True,
            temperature=0.7,
            top_p=0.9,
            pad_token_id=tokenizer.eos_token_id
        )

    input_token_length = inputs.input_ids.shape[1]
    generated_tokens = outputs[0, input_token_length:]
    response_text = tokenizer.decode(generated_tokens, skip_special_tokens=True)

    print(f"   Model Response: {response_text.strip()}")
    return ChatResponse(response=response_text.strip())
```

Listing A.1: FastAPI Chat Endpoint

A.2 Flutter Chat Message Sending Logic (chat_screen.dart)

This code snippet in Dart of the Flutter application describes the act of sending messages, the function of which is called `sendMessage`. The role of this function is to process the input of the user, generate a message object, update the UI, handle the loading state, and invoke the API service.

```dart
void _sendMessage() async {
    final userMessageText = _messageController.text.trim();
    if (userMessageText.isEmpty) return;

    final currentUser = FirebaseAuth.instance.currentUser;
    if (currentUser == null) return;

    final userMessage = ChatMessage(
        text: userMessageText,
        isUser: true,
        timestamp: DateTime.now(),
        uid: currentUser.uid);

    _addMessage(userMessage, saveToDb: false);
    _messageController.clear();
    if (mounted) setState(() => _isLoading = true);

    try {
      String currentConversationId;

      if (_conversationId == null) {
        currentConversationId =
            await _chatService.createNewConversation(userMessageText);
        if (mounted)
            setState(() => _conversationId = currentConversationId);
      } else {
        currentConversationId = _conversationId!;
      }

      await _chatService.sendMessage(
          currentConversationId, userMessageText, isUser: true);

      final historyForApi = _messages.reversed
          .map((msg) => {
              'role': msg.isUser ? 'user' : 'assistant',
              'content': msg.text
           })
          .toList();

      final botResponseText =
          await _apiService.getChatbotResponse(userMessageText, historyForApi);

```

```
      final botMessage = ChatMessage(
          text: botResponseText,
          isUser: false,
          timestamp: DateTime.now(),
          uid: "assistant");

      _addMessage(botMessage, saveToDb: false);
      await _chatService.sendMessage(
          currentConversationId, botResponseText, isUser: false);

    } catch (e) {
      print("Error in sending message: $e");
    } finally {
      if (mounted) setState(() => _isLoading = false);
    }
}
```

Listing A.2: Flutter Send Message Logic

Appendix B

User Acceptance Testing Survey

The questions presented to the five volunteers during the User Acceptance Testing (UAT) phase were as follows as indicated in Chapter 6. Participants were required to rate the statements on a scale of 1 (Strongly Disagree) to 5 (Strongly Agree) and they were also requested to comment freely.

B.1 Survey Questions

1. **Onboarding:** The account creation and log in process was easy and uncomplicated.
2. **Usability:** I was not going through a complicated application and got to know how to use its features.
3. **Interface Design:** The user interface (colors, fonts, layout) felt clean and calming.
4. **Response Quality:** The answers of the chatbot were appropriate to my questions and messages.
5. **Empathy:** I felt that the chatbot's responses were empathetic and supportive.
6. **Performance:** The app was responsive and seemed quick, and I did not experience any irritating delays.
7. **Trust and Security:** I believe that the information on my conversations is being managed privately and safely through this application.
8. **Overall Experience:** I would consider using this application again in the future for mental well-being support.
9. **Open Feedback:** Do you have any additional comments or suggestions for improvement?

References

[1] Fitzpatrick, K., Darcy, A., & Vierhile, M. (2017). Delivering Cognitive Behavior Therapy using a Conversational Agent: Evaluation and Feasibility. *JMIR Mental Health*, 4(2), e19. Cited on p. 5.

[2] Inkster, B., Sarda, E., & Subramanian, V. (2018). An Empathy-Driven, Conversational Artificial Intelligence Agent (Wysa) for Digital Mental Well-Being: Real-World Data Evaluation Mixed-Methods Study. *JMIR mHealth and uHealth*, 6(11), e12106. Cited on p. 5.

[3] Mehta, A., et al. (2021). Evidence-Based Mental Health Support Through an AI-Powered Chatbot (Youper). *Frontiers in Psychiatry*, 12, 695. Cited on p. 5.

[4] Hollis, C., et al. (2017). Annual Research Review: Digital health interventions for children and young people with mental health problems. *Journal of Child Psychology and Psychiatry*, 58(4), 474–503. Cited on p. 5.

[5] OpenAI. (2023). GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*. Cited on p. 5.

[6] Carlini, N., et al. (2021). Extracting Training Data from Large Language Models. *USENIX Security Symposium*, 2633–2650. Cited on pp. 6 and 7.

[7] Touvron, H., et al. (Meta AI). (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*. Cited on p. 6.

[8] Almazrouei, E., et al. (Technology Innovation Institute). (2023). The Falcon Series of Open Language Models. *arXiv preprint arXiv:2311.16867*. Cited on p. 6.

[9] Jiang, A. Q., et al. (Mistral AI). (2023). Mistral 7B. *arXiv preprint arXiv:2310.06825*. Cited on p. 6.

[10] Hu, E. J., et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations (ICLR)*. Cited on p. 6.

[11] Liu, Y., et al. (2023). Mental-LLM: Leveraging Large Language Models for Mental Health Prediction via Online Text Data. *arXiv preprint arXiv:2307.12921*. Cited on pp. 6, 7, and 8.

[12] Dettmers, T., et al. (2024). QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems (NeurIPS)*. Cited on p. 7.

[13] Kaissis, G. A., et al. (2020). Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6), 305–311. Cited on p. 7.

[14] Xu, M., et al. (2023). A Survey on Edge Intelligence: Emerging Trends and Challenges in Large Language Models. *IEEE Internet of Things Journal*. Cited on pp. 7 and 8.

[15] European Commission. (2016). Regulation (EU) 2016/679 (General Data Protection Regulation). *Official Journal of the European Union*. Cited on p. 6.

[16] Google. (2024). *Firebase Documentation*. https://firebase.google.com/docs Cited on p. 8.