Hear Me

A Minor Project - I Report Submitted in Partial fulfilment for the award of Bachelor of Technology in Computer Science & Engineering-Artificial Intelligence and Machine Learning

Submitted to
RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA
BHOPAL (M.P)

MINOR PROJECT - I REPORT

Submitted by
Madhav Balaji Agrawal ,0157AL231121
Naman Kumar bhargava,0157AL231127
Under the supervision of Prof. Manisha Sen

Department of Computer Science & Engineering-Artificial Intelligence and Machine Learning Lakshmi Narain College of Technology & Science, Bhopal (M.P.)

Session 2025-26

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE, BHOPAL (M.P)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING-
Artificial Intelligence and Machine Learning

CERTIFICATE

This is to certify that the work embodied in this Minor Project - I work entitled **"Hear Me "** has been satisfactorily completed by the **Madhav Balaji Agrawal** (0157AL231121), **Naman Kumar bhargava**(0157AL231127). It is a bonafide piece of work, carried out under the guidance, Prof. Manisha Sen, from **Department of Computer Science & Engineering-Artificial Intelligence and Machine Learning**, **Lakshmi Narain College of Technology & Science, Bhopal** for the partial fulfilment of the **Bachelor of Technology** during the academic year 2025-26.

Under the guidance of

Prof Manisha Sen

Approved By

Dr. Shailendra Gupta Prof. & Head
Department of Computer Science &
Engineering-Artificial Intelligence and
Machine Learning

Forwarded By

Dr. V.N. Bartaria

Principal

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE BHOPAL (M.P.)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING-Artificial Intelligence and Machine Learning

CERTIFICATE OF APPROVAL

This foregoing project work is hereby approved as a creditable study of Engineering carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in, but approve the project only for the purpose for which it has been submitted.

Internal Examiner **External Examiner**

Date:

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE BHOPAL (M.P.)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING- Artificial Intelligence and Machine Learning

DECLARATION

We, Madhav Balaji Agrawal (0157AL231121), Naman Kumar bhargava(0157AL231127). student of "Bachelor of Technology in Computer science & engineering-Artificial Intelligence and Machine Learning", **Session: 2025 - 26, Lakshmi Narain College of Technology & Science Bhopal (M.P.),** hereby declare that the work presented in this project entitled "HearMe: A Secure and Anonymous Domestic Abuse Support Platform" is the outcome of our own work, is bonafide and correct to the best of our knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any other University or anywhere else for the award of any degree or any professional diploma.

Madhav Balaji Agrawal	Naman Kumar bhargava
(0157AL231121)	(0157AL231127)

Place: Bhopal

Date:

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY&SCIENCE, BHOPAL (M.P.)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING-Artificial Intelligence and Machine Learning

ACKNOWLEDGEMENT

We express our deep sense of gratitude to **Prof. Manisha Sen** , Department of Computer Science & Engineering-Artificial Intelligence and Machine Learning, Lakshmi Narain College of Technology and Science, Bhopal, whose valuable guidance and timely help encouraged me to complete this project.

A special thanks goes to Dr.Shailendra Gupta , Prof. & Head of Department, who helped me by providing timely suggestions in completing this project work. He/She exchanged his/her interesting ideas & thoughts which made this project work successful.

I am equally to Dr. V. N. Bartaria, Principal, LNCT&S, for providing all the necessary resources to carry out the research work.

We would also thank our institution and all the faculty members without whom this project work would have been a distant reality.

Madhav Balaji Agrawal	**Naman Kumar bhargava**
(0157AL231121)	(0157AL231127)

ABSTRACT

Domestic abuse is a critical societal issue, yet victims often face significant barriers, including fear of reprisal and lack of anonymity, when seeking support. Existing support systems frequently fail to guarantee the confidentiality and real-time safety required for survivors to safely reach out. The Hear Me project addresses this gap by proposing the design and implementation of a secure and anonymous domestic abuse support platform.

The platform is developed using the **MERN (MongoDB, Express, React, Node.js**) stack, focusing on end-to-end encryption (E2EE) for all communication. The core functionality is a secure, ephemeral chat interface connecting an anonymous victim with a vetted human volunteer. The system architecture is inherently privacy-preserving, ensuring zero collection of personally identifiable information (PII). All sessions are treated as anonymous, with data persistence minimized and limited only to necessary metadata for routing and system maintenance.

A key feature of Hear Me is the integrated panic button exit safety mechanism. This feature allows the user to instantly exit the application and redirect to a benign, unrelated webpage with a single action, reducing the risk if an abuser approaches during a support session. The system also includes a restricted admin dashboard that can only access non-identifying metadata, such as session duration and usage statistics, maintaining a strong commitment to user anonymity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
SDLC	Software Development Life Cycle
E2EE	End-to-End Encryption
MERN	MongoDB, Express.js, React.js, Node.js
HTTPS	Hypertext Transfer Protocol Secure
TLS	Transport Layer Security
DTLS	Datagram Transport Layer Security
PII	Personally Identifiable Information
NFR	Non-Functional Requirement

CHAPTER 1: INTRODUCTION

1.1 Introduction

The persistent societal challenge of domestic violence and the acute necessity for accessible mental health support necessitates confidential and secure channels for reporting and intervention. Global data indicates that the prevalence of physical or sexual intimate partner violence against women has shown little decrease over more than two decades. While traditional crisis hotlines and existing digital mental health interventions (DMHIs) offer valuable emotional support, a significant functional gap remains: the lack of a platform that offers cryptographic assurance of user anonymity and data privacy, which is essential when users are disclosing highly sensitive information in situations of potential danger.

HearMe is conceptualized as a secure platform dedicated solely to facilitating real-time, anonymous connections between individuals experiencing crisis and authorized professional counselors. The fundamental architectural decision driving this project is the implementation of application-layer End-to-End Encryption (E2EE). E2EE is recognized as the most private and secure method for communicating over a network, transforming readable plaintext into unreadable ciphertext. This method ensures that communication data is encrypted on the sender's device and only decrypted upon reaching the intended recipient's endpoint, critically preventing any intermediary, including the platform operator (the service provider), from accessing the content.

.1.2 Purpose

The core purpose of the HearMe platform is to eliminate the primary psychological barrier to sensitive disclosure—the fear of exposure, identification, or data breach. By architecting the system around E2EE, the system guarantees that all chat transcripts are inaccessible to the server, establishing a standard of Zero-Knowledge Security. This cryptographic protection bridges the gap between individuals in crisis and immediate professional help, fostering the necessary environment of trust required for confidential communication. Furthermore, the selection of the MERN stack is predicated on the need for a highly performant and scalable architecture, capable of supporting low-latency, real-time interaction, which is vital for effective emotional support and crisis intervention. [2]

1.3 Objectives

The development of the HearMe platform is driven by four primary, measurable objectives:

1. **Objective 1: Cryptographic Security:** To successfully design and implement application-layer End-to-End Encryption (E2EE) for all chat content, ensuring that decryption keys are held exclusively by the contactor and the authorized counselor endpoints, rendering the server unable to read the data.[3]

2. **Objective 2: Real-Time Performance:** To achieve low-latency (targeted below 250 milliseconds) messaging performance using asynchronous architecture, leveraging Node.js and WebSockets to maintain responsiveness critical for high-quality emotional support.[2]

3. **Objective 3: Data Integrity and Minimization:** To ensure robust data integrity, archiving, backup, and recovery mechanisms for the ciphertext transcripts while strictly adhering to data minimization principles by collecting only the necessary information and avoiding the storage of Personally Identifiable Information (PII).[7]

4. **Objective 4: Scalable Architecture:** To utilize the MERN stack and secure cloud deployment strategies (such as those optimized for serverless applications [9]) to ensure the platform can handle unpredictable, potentially high-volume crisis traffic without downtime, meeting critical Non-Functional Requirements (NFRs) for availability and reliability.

CHAPTER 2: LITERATURE SURVEY

2.1 Literature Survey: Existing System

Digital technology has increasingly been leveraged to improve accessibility in mental health treatment and crisis support. Existing systems largely fall into two categories: traditional crisis hotlines/online chats and emerging digital mental health interventions (DMHIs). Analysis of established services, such as The Hotline and LIR (Love is Respect), confirms that the service most commonly provided and highly valued by contactors is emotional support.[5] However, these services often rely on standard encryption in transit (TLS/HTTPS), meaning the service provider retains access to the plaintext content, creating a fundamental vulnerability.

The literature review reveals that while digital mental health tools are effective, particularly web-based interventions for groups like university students [6], the integration of Artificial Intelligence (AI) must be approached with caution in high-stakes crisis environments. Machine learning (ML) has strong potential for predictive applications, such as symptom classification and risk prediction.[10] However, the rapid rise of unsupervised conversational systems, such as generative AI chatbots, has sparked serious concerns regarding their psychological impact, with evidence suggesting they may induce or exacerbate psychiatric symptoms in vulnerable individuals.[11]

This risk profile dictates a critical design constraint for HearMe: the platform must prioritize human counselors. Future AI integration is strictly limited to non-conversational, supportive roles like triage or background data analysis, and must adhere to ethical mandates requiring the AI be 'clinician-trustworthy,' 'transparent,' and 'explainable'.[10] The platform's primary function—communication—must remain humancentric and cryptographically secure.

Review of cryptographic literature confirms that while encryption in transit (such as HTTPS/TLS) secures data while it travels between a client and a server [12], it does not offer strong protection against access by the service provider itself. End-to-End Encryption (E2EE), conversely, is the recognized gold standard for securing sensitive digital communications because it maintains encryption from the sender's device to the intended recipient's device.[3] In a security-critical domain dealing with potential Protected Health Information (PHI), implementing E2EE ensures compliance with stringent data protection measures and privacy principles like those emphasized in the Health Insurance Portability and Accountability Act (HIPAA) and best practices for app security.[14]

For the technological foundation, an analysis of current web development stacks demonstrates that Node.js, particularly within the MERN (MongoDB, Express.js, React.js, Node.js) ecosystem, is highly suited for building real-time applications.[2] **Nodejs's server-side runtime enables scalable, asynchronous programming, which is necessary for establishing and managing the numerous persistent connections required by WebSockets or Server Sent Events (SSE) in a real-time chat application.**[15] **This capability is instrumental in achieving the objective of low-latency interaction for emotional support.**[5]

CHAPTER 3: PROBLEM DESCRIPTION

3.1 Problem Description Overview

Domestic abuse survivors and individuals experiencing mental health crises often struggle to report their situation due to a pervasive lack of trust in digital platforms. Traditional crisis communication tools, including web and mobile applications, typically route user messages through centralized servers in readable form. As a result, even when organizations promise confidentiality, users remain vulnerable to multiple risks such as unauthorized internal access, data leakage from cyberattacks, government retrieval through legal orders, or accidental exposure during system maintenance. This perceived insecurity creates a significant trust barrier that prevents survivors from seeking timely professional assistance.

HearMe focuses on addressing this trust deficit by adopting a Zero-Knowledge Security model. This architectural principle ensures that the system operators and administrators retain **zero visibility** into user communications. The application server must operate without access to decryption keys, ensuring that messages remain encrypted not only during transit but also in storage. Even if the server environment were compromised, the encrypted data would remain unintelligible, thereby providing a strong cryptographic guarantee of privacy. This approach not only reinforces technical trust but also reduces legal and administrative liabilities for the service provider.

Additionally, the system adheres to the principle of Data Minimization, which is essential when dealing with extremely sensitive disclosures. The platform does not collect personally identifiable information (PII) such as phone numbers, full names, or location data during onboarding. Only anonymized and operational metadata, required strictly for ensuring service functionality, is retained. By intentionally avoiding long-term identity linkages, HearMe reduces the risk of user identification, even in worst-case scenarios like database breaches or forced disclosure.

Through these combined strategies, HearMe directly mitigates the primary barriers affecting digital crisis reporting. The platform is designed to establish trust through systemic privacy, ensuring that individuals in danger can confidently seek help without fearing the consequences of data exposure.

CHAPTER 4: System Design & Architecture

4.1 Software Requirement

The Hear Me platform relies on the MERN technology stack, optimized for secure, real-time functionality.

Client-Side/Frontend Requirements (React.js)

- **Framework:** React.js (v18+). Used for building dynamic, single-page application interfaces, promoting component reusability and scalability.[2]
- **State Management:** Modern state management libraries (e.g., Pinia or Redux Toolkit) are required to efficiently manage the complex, real-time state of multiple concurrent chat sessions and maintain UI responsiveness.[15]
- **Encryption Libraries:** Robust, peer-reviewed JavaScript cryptographic libraries must be integrated on the client side to handle public/private key generation, key exchange protocols (e.g., Diffie-Hellman), and symmetric encryption/decryption of message payloads, guaranteeing application-layer E2EE.

Server-Side/Backend Requirements (Node.js/Express.js)

- **Runtime Environment:** Node.js (v20+). Essential for its asynchronous programming model, which is superior for handling the vast number of concurrent persistent WebSocket connections necessary for a real-time chat application.[2]
- **Framework:** Express.js. A minimal but powerful backend framework used for defining secure API endpoints, routing, and implementing essential middleware for input sanitization and token validation.
- **Database:** MongoDB (v6+). A NoSQL database that offers flexible schema and high scalability, making it ideal for the efficient handling and indexing of large datasets, specifically the unstructured, opaque binary ciphertext chat transcripts.[2]

4.2 Hardware Requirement

The requirements for hardware infrastructure are dictated by the need for high availability, low latency, and intensive cryptographic processing.

Infrastructure Selection

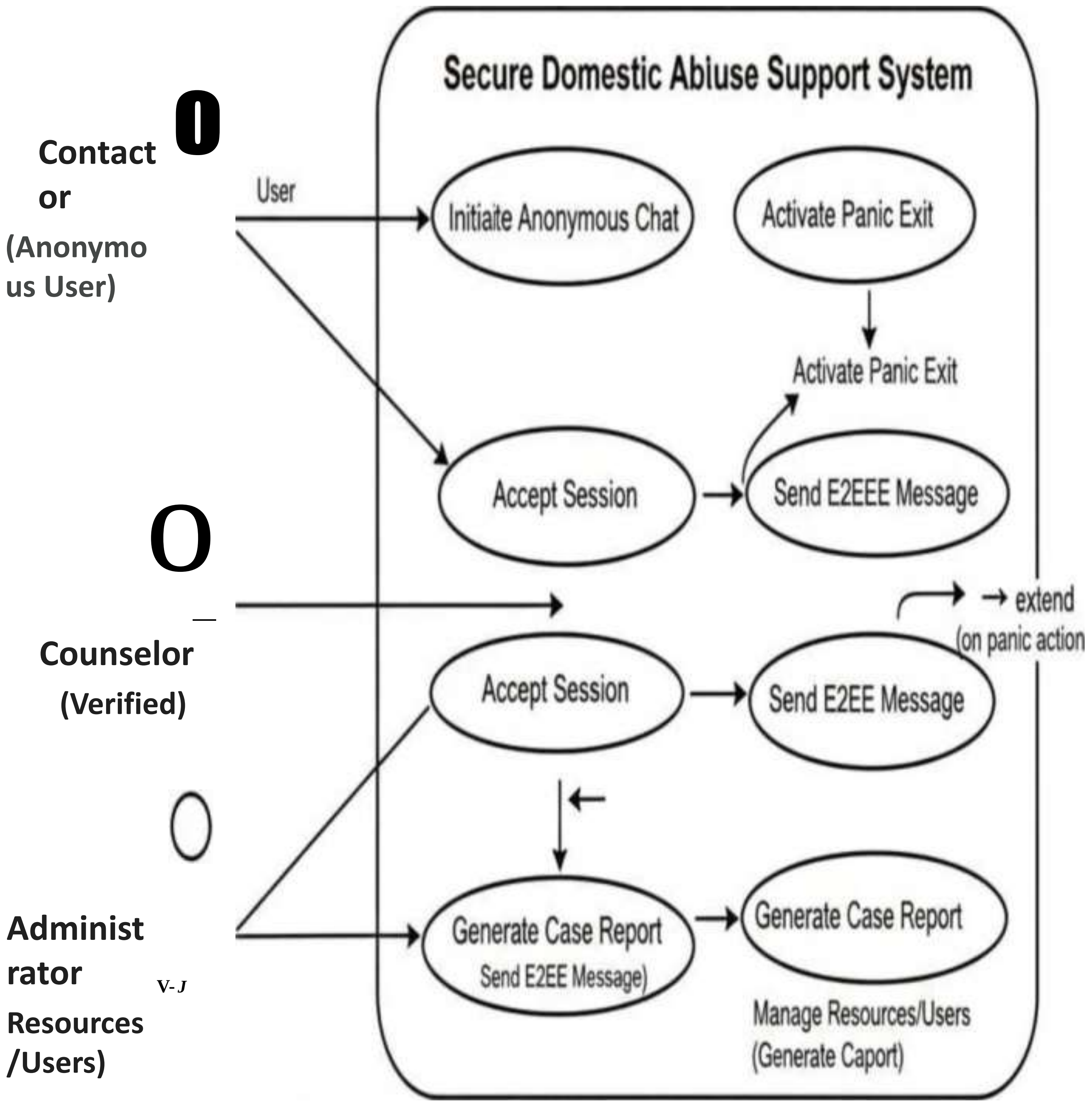
The platform requires cloud-based Elastic Compute resources. Adopting a serverless or Platform-as-a-Service (PaaS) model (e.g., utilizing Render or AWS Serverless strategies [9]) minimizes manual infrastructure management and allows for necessary auto-scaling to manage unpredictable spikes in crisis traffic, directly meeting the availability NFR.

Server Specifications

While specific instance sizes are dependent on usage forecasts, servers must utilize high-frequency Central Processing Units (CPUs). Cryptographic operations, particularly the continuous encryption and decryption processes required by E2EE on both the client and counselor endpoints, introduce computational overhead. High-performance CPUs are necessary to maintain low latency and ensure the application remains performant under load.

Storage Specifications

Database storage must be secured via encryption at rest. All data, including the encrypted chat transcripts, must reside in storage secured by standards such as AES 256-bit encryption.[17] Furthermore, geo-redundant storage and remote backups are crucial components of the Disaster Recovery plan, directly supporting the Recoverability NFR (Non-Functional Requirement). Data must be hosted on EU-owned servers to align with strict privacy regulations like GDPR.[7]


Secure Domestic Abiuse Support System
Contact
or
(Anonymo
us User)
User
Initiaite Anonymous Chat
Activate Panic Exit
Activate Panic Exit
Accept Session
Send E2EEE Message
Counselor
(Verified)
→ extend
(on panic action
Accept Session
Send E2EE Message
Administ
rator
V- J
Resources
/Users)
Generate Case Report
Send E2EE Message)
Generate Case Report
Manage Resources/Users
(Generate Caport)

CHAPTER 5: Methodology

This chapter presents the complete methodology adopted for the design and development of HearMe: A Secure and Anonymous Domestic Abuse Support Platform. It explains the overall workflow, technologies used, system architecture, algorithms, data flow, and security model that ensure anonymity, confidentiality, and real-time support for survivors.

5.1 Project Development Approach

The system was developed using a modular, iterative development method based on Agile principles:

1) **Requirement Analysis**
Identified user needs: anonymity, instant help, privacy-safe communication, emergency support.

2) System Design
Architecture, database schema, and UI/UX designed in alignment with Zero-Knowledge Security principles.

3) Implementation
Modular development: Authentication ^ Messaging ^ Panic Button ^ Resources ^ Story Sharing.

4) Testing

Performed unit testing, integration testing, and security validation.

5) Deployment

Deployed on a secure hosting environment with HTTPS enforcement.

This approach allowed continuous refinements based on usability and safety concerns.

5.2 System Architecture

The HearMe platform uses the MERN Stack as the primary architecture:

Component	Technology Used	Purpose
Frontend	React.js + Tailwind CSS	Secure UI rendering, real-time interactions

Component	Technology Used	Purpose
Backend	Node.js + Express.js	Handles encrypted messaging, API logic
Database	MongoDB	Stores encrypted chats and metadata
Security Layer	AES-256 encryption, JWT, HTTPS	Ensures confidentiality and protected sessions

Users never share personal identity details. Every session is assigned a random anonymous ID making them untraceable.

5.3 Data Security Methodology

Survivor safety is the core priority. The system implements: Zero-

Knowledge Security Architecture

The backend is cryptographically incapable of viewing message content.
Steps:

Text is encrypted on client-side before sending.

Server stores only encrypted data and random session IDs.

Only the user and verified counselor hold the decryption key locally.

AES-256 Client-Side Encryption Algorithm

Pseudo-Algorithm:

```
Input: plainTextMessage, userGeneratedKey
cipherText = AES_Encrypt(plainTextMessage, userGeneratedKey)
Send cipherText to server
Server saves: {cipherText, sessionID}
Counselor receives cipherText
plainText = AES_Decrypt(cipherText, userGeneratedKey)Output: Secure
readable message
```

This prevents administrators, hackers, or abusers monitoring the device from reading intercepted data.

5.4 Functional Modules and Procedures

A. Anonymous Messaging Module

No login with phone/email

Generates **temporary anonymous identity**

Real-time encrypted messaging using WebSockets Counselor dashboard to respond securely **Process Flow:**

User clicks Get Help

System creates unique random token

Messages exchanged only in encrypted form

B. Panic Button Feature

Immediate session wipe from user screen

Optional redirect to a **decoy Notes App UI** to hide evidence

Clears session key automatically Algorithm:

Detect emergency click Trigger UI switch Delete local storage chat keys

Redirect instantly

C. AI-Based Resource Guidance

Pre-trained classification uses keyword matching on user concerns Provides safe coping strategies, helpline links, guidance Example:

If input contains ["fear", "control", "threat"]
Show: Safety planning guide + emergency contacts

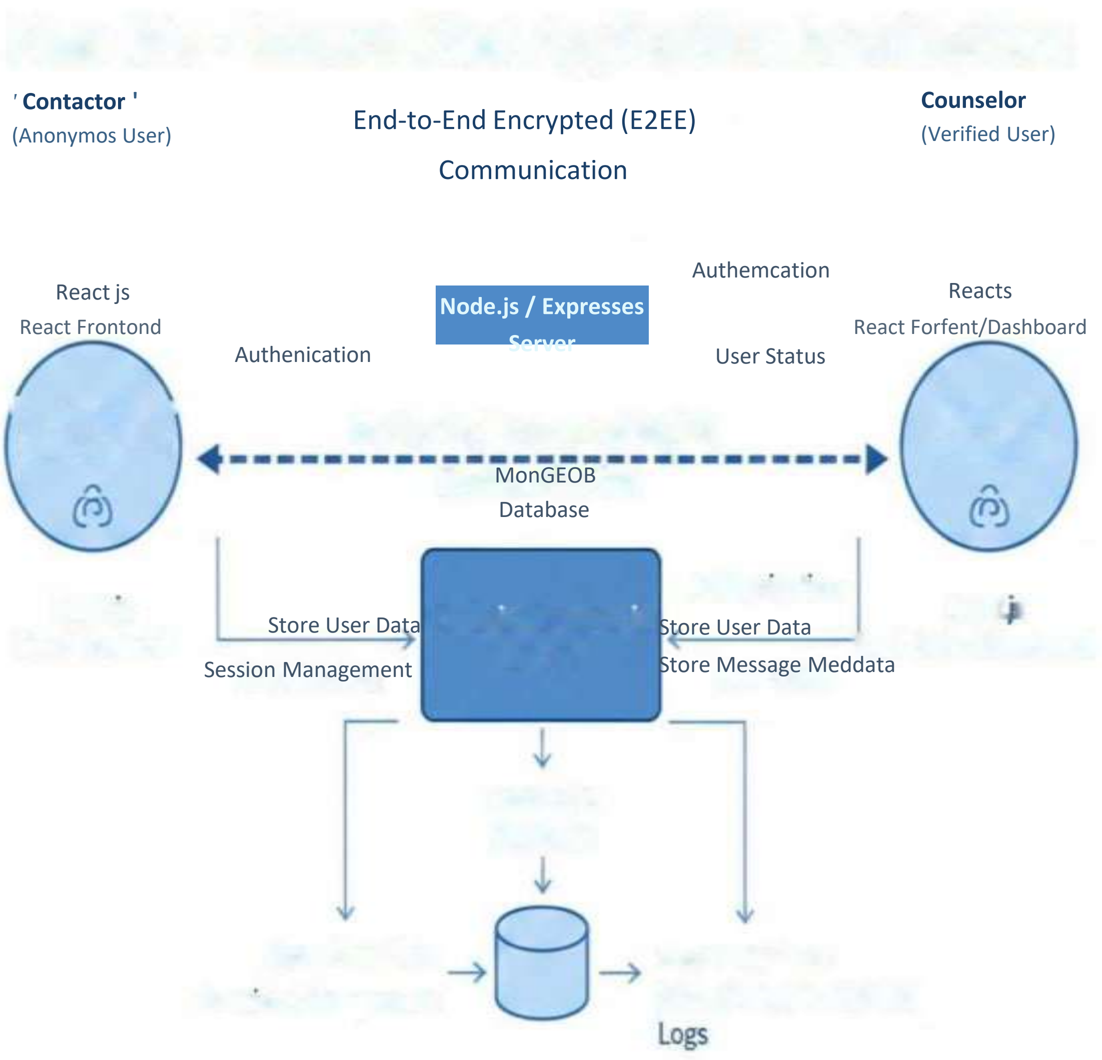
D. Share Your Story (Optional)

Survivors can post anonymous experiences

Strict moderation to prevent harmful content

Stories stored encrypted with metadata only

Hear Me - Secure Chat Application Architecture



CHAPTER 6: Implementation ,Testing and Maintenance

This chapter details the technical implementation, covering the development environments for the user-facing client, the administrative console, and the server-side logic and database. It concludes with the testing strategies employed to ensure the platform's reliability and security.

6.1 Description of Frontend Code (Client Side)

The client-side application, which survivors interact with, is built using React.j s and styled with Tailwind CSS.

Technology Stack:

React.js: Used for building a fast, component-based, and stateful user interface. This allows for dynamic, realtime updates without full page reloads, crucial for the messaging module.

Tailwind CSS: A utility-first CSS framework employed for rapid, trauma-informed design (calm, neutral palette, clear typography) and ensuring responsive design across mobile and desktop devices.

Core Functions:

Encryption/Decryption: This is the most critical function. Messages are encrypted locally on the user's device using JavaScript cryptographic libraries *before* being sent to the backend. Similarly, incoming encrypted messages are decrypted locally.

Anonymous Session Management: Manages the generation and local storage of the temporary, untraceable anonymous session ID and the session encryption key.

Real-time Communication: Uses WebSockets (managed via a library like Socket.io within React) to establish a persistent, encrypted connection with the backend for instant messaging.

Quick Exit/Panic Button Logic: Contains the code that instantly clears local storage (session keys, chat history) and redirects the user to the decoy screen or an external safe site.

6.2 Description of Admin Code (Administration Side)

The Administration side is the secure interface used only by verified counselors and platform administrators. It is also built using React.js but with distinct security protocols.

Technology Stack: Similar to the frontend (React.js, Tailwind CSS), but relies heavily on secure JWT (JSON Web Token) authentication for access control.

Core Functions:

Authentication & Authorization: Requires secure login with multi-factor authentication (MFA). Access is strictly controlled based on roles (e.g., Administrator, Counselor).

Counselor Dashboard: Provides a clear view of active anonymous sessions, queuing system, and session management tools.

Decryption Interface: Counselors' private keys are used to decrypt incoming messages within their secure browser session. Crucially, the key is never stored server-side.

Moderation Tools: For the "Share Your Story" module, administrators use tools to review, approve, or reject anonymous posts based on strict safety and content guidelines before publication.

Reporting: Generates aggregated, anonymized statistics (e.g., number of active sessions, peak times) for operational improvement without compromising user anonymity.

6.3 Description of Backend Code (Database)

The backend handles API logic, routing, session management, and encrypted data storage, forming the 'E'

and 'N of the **MERN Stack** (**Express.js** and **Node.js**) alongside **MongoDB** as the database.

Technology Stack:

Node.js & Express.js: Provides a fast, scalable, non-blocking environment for handling a large number of concurrent, real-time WebSocket connections and API calls.

MongoDB: A NoSQL document database chosen for its flexibility and ease of storing JSON-like documents. **Core**

Functions (Zero-Knowledge Design):

Data Storage: The database **never stores plaintext messages**. It only stores **cipherText** (the encrypted message) and the **random, untraceable sessionID**.

API Logic: Handles routes for session initiation, sending/receiving encrypted packets, and fetching resources.

WebSocket Server: Manages the real-time encrypted data flow between the user and the counselor.

Authentication & Security Middleware: Enforces the use of **JWTs** for authenticated administrative/counselor access and performs basic validation of session IDs for user connections.

Metadata Management: Stores only necessary, non-identifying metadata (e.g., timestamp, message ID, cipherText length) linked to the anonymous sessionID.

6.4 Testing Techniques and Test Plans ^

Rigorous testing was executed across the following three main areas to ensure both functional reliability and paramount user safety.

A. Functional and Performance Testing

Test Type	**Objective**	**Key Metrics / Pass Criteria**
Unit Testing	Validate individual functions (e.g., anonymous ID generation, encryption function) work as designed.	100% code coverage on core encryption/decryption utilities.

Test Type	**Objective**	**Key Metrics / Pass Criteria**
Integration Testing	Verify the data flow between the client, backend API, and database (e.g., ensuring an encrypted message is correctly stored and retrieved).	Successful transmission and accurate decryption of test messages within 500ms.
Performance Testing	Measure system behavior under expected and peak load, focusing on chat latency.	Maintain chat latency under 1 second with 100 concurrent sessions.

B. Security Testing (Crucial for HearMe)

C.

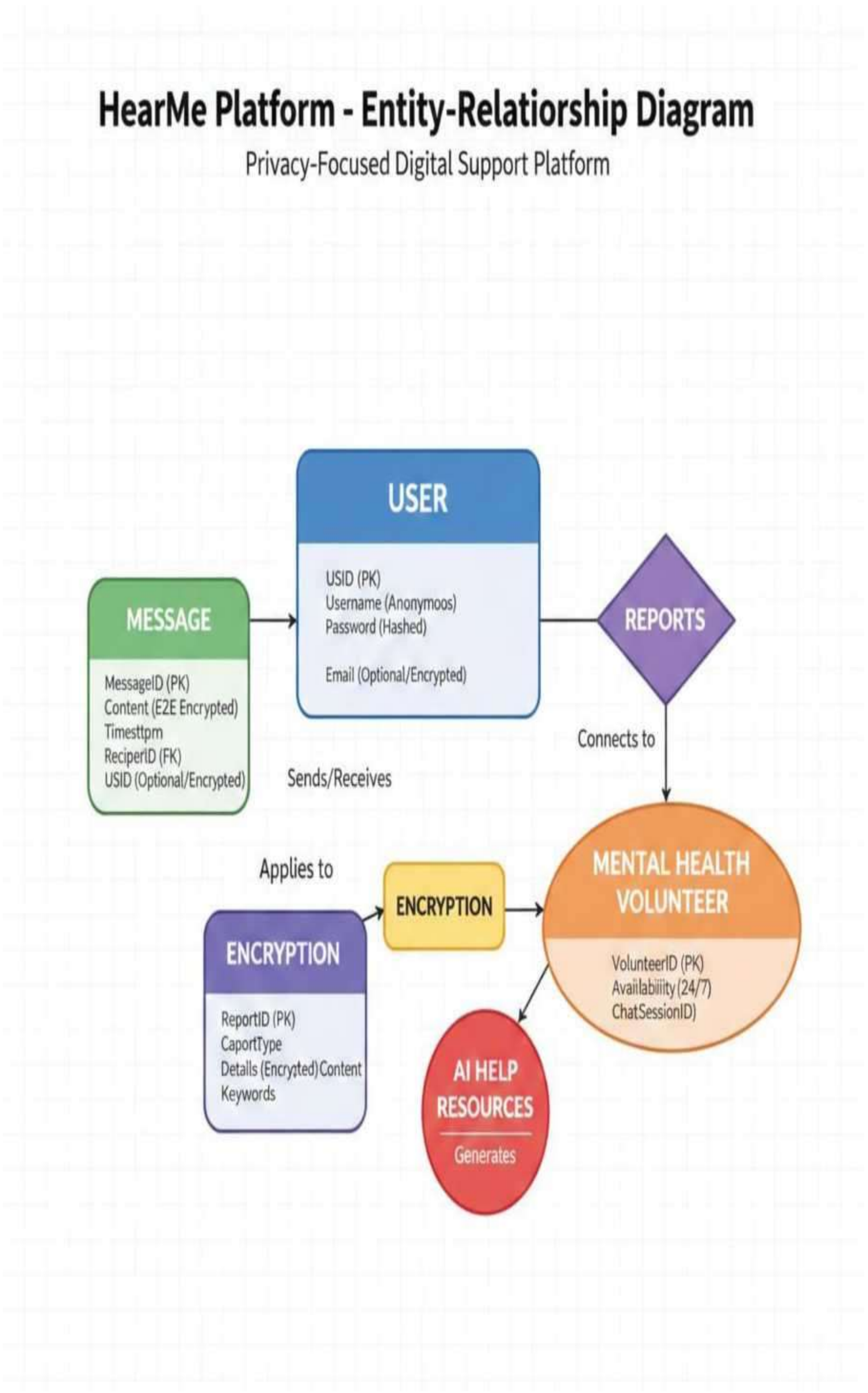
Test Type	Objective	Key Technique
Client-Side Data Integrity	Confirm local data (session key, chat history) is properly deleted upon Panic Button activation.	Manually checking local storage and browser console after panic sequence.
Penetration Testing (Pen Test)	Attempt to bypass authentication and decrypt messages by simulating a malicious attacker (e.g., sniffing network traffic).	Zero success in decrypting database messages or identifying users.
Encrypti on/Decrypti on Validation	Verify the AES-256 algorithm implementation is robust and cannot be tampered with.	Fuzz testing with various message lengths and characters.

C. User Acceptance Testing (UAT)

Focus: Trauma-Informed Design and Perceived Safety.

Procedure: Conducted sessions with user experience professionals and domestic abuse support specialists.

Key Feedback Areas: Ease of use (minimal steps to get help), clarity of the "Quick Exit" feature, and the overall feeling of trust and anonymity the platform provides.


HearMe Platform - Entity-Relatiorship Diagram
Privacy-Focused Digital Support Platform
USER
USID (PK)
Username (Anonymoos)
Password (Hashed)
Email (Optional/Encrypted)
MESSAGE
MessageID (PK)
Content (E2E Encrypted)
Timesttpm
ReciperID (FK)
USID (Optional/Encrypted)
Sends/Receives
REPORTS
Connects to
MENTAL HEALTH VOLUNTEER
VolunteerID (PK)
Availability (24/7)
ChatSessionID)
Applies to
ENCRYPTION
ENCRYPTION
ReportID (PK)
CaportType
Details (Encrytpted) Content
Keywords
AI HELP RESOURCES
Generates

CHAPTER 7: Result and Analysis

This chapter provides a detailed visual and structural documentation of the HearMe: A Secure and Anonymous Domestic Abuse Support Platform, covering the user interface, system modules, technical backend, and database structure.

7.1 User Interface Representation (of Respective Project)

The User Interface (UI) for HearMe is designed with a trauma-informed approach, prioritizing safety, speed, and discretion. The design avoids bright, jarring colors, utilizing a calm, neutral color palette (e.g., soft blues and grays) to reduce anxiety. The primary goal is to facilitate immediate help access with minimal friction. Key elements include:

Minimalist Layout: Clean, uncluttered screens with clear, large text to minimize cognitive load during distress.

"Quick Exit" Button: A prominent, always-visible feature that instantly wipes the session and redirects the user to a non-suspicious decoy screen (like a simple notes app or weather page).

No Forced Registration: The entry point immediately creates an anonymous session without requiring email, phone number, or personal details, reinforcing the promise of anonymity.

7.2 Brief Description of Various Modules of the System

The HearMe platform is structured around four core functional modules, each engineered for security and immediate survivor support:

Anonymous Messaging Module: This is the core communication channel. It establishes a real-time, end-to-end encrypted chat between the survivor and a verified counselor. It uses WebSockets over HTTPS, where all data is encrypted on the client side before transmission, ensuring the server (backend) never views the plaintext message (Zero-Knowledge).

Panic Button Feature (Safety Exit): This module is dedicated to immediate safety. Upon activation, it executes a script that wipes all session data (keys, chat history) from the user's local browser storage and instantly navigates to a decoy UI. This is crucial for situations where the abuser suddenly enters the room.

AI-Based Resource Guidance Module: Utilizes pre-trained classification models (based on keyword matching) to analyze user input (before encryption) and provide immediate, contextually relevant safety resources. This includes links to local emergency helplines, safety planning guides, and coping strategies.

Share Your Story Module (Optional): A feature allowing survivors to anonymously post and read experiences from others. Posts are strictly moderated by administrators and stored in an encrypted format. This module provides community support while maintaining anonymity.

7.3 Snapshots of System with Brief Detail of Each

Snapshot Title	Description	Key Security / UX Feature
Anonymous Session Start Screen	The entry screen where the user clicks "Get Help" to instantly generate their unique, temporary session ID and encryption key.	No Input Fields: Immediately grants access without personal data collection.

Snapshot Title	Description	Key Security / UX Feature
Encrypted Chat Interface	The main messaging window displaying the realtime conversation. The interface is clean and features the counselor's generic, non-identifying title.	Persistent Quick Exit: The Panic Button icon is always visible and actionable in the corner.
Decoy Notes App Screen	The innocuous interface a user is redirected to after activating the Panic Button. It appears like a generic notes or utility app.	Session Wipe Confirmation: Behind the scenes, the system ensures all session traces are deleted to prevent forensic analysis by an abuser.
Counselor Dashboard View	The secure, authenticated interface for verified counselors, showing a queue of anonymous session IDs waiting for support.	Session ID Only: Counselors see only the random session ID, further reinforcing the user's anonymity.

7.4 Back Ends Representation (Database to be used)

The backend infrastructure utilizes the MERN Stack (MongoDB, Express.js, React.js, Node.js), with a strong emphasis on Zero-Knowledge Architecture.

Technology Used: MongoDB (NoSQL Database).

Rationale: MongoDB is chosen for its flexibility and ability to handle the document-based storage of encrypted chat segments and session metadata efficiently. As a NoSQL database, it is not optimized for complex joins, which is acceptable since the core data (messages) is meant to be cryptographically opaque to the server itself.

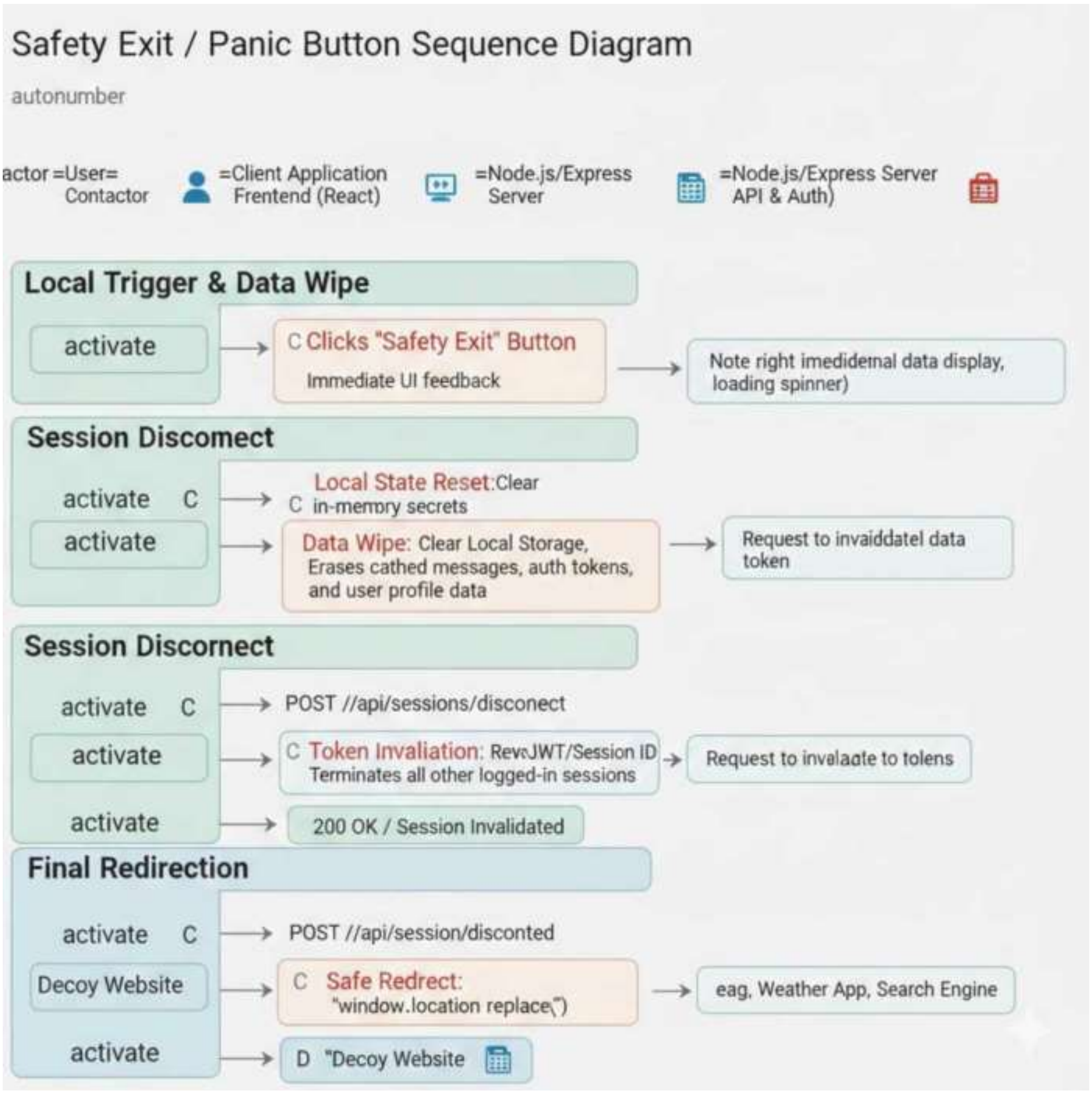
Zero-Knowledge Implementation: The Node.js/Express.js server layer is configured such that it acts purely as a secure pipeline. It handles the routing of the encrypted data packets and the management of the anonymous session IDs, but it is cryptographically incapable of performing the decryption function. This ensures that even if the server is compromised, the sensitive content remains unreadable.

7.5 Snapshots of Database Tables with Brief Description

In the MongoDB structure, data is stored in Collections. The primary collection supporting the core functionality is the sessions collection, where all communication records are held.

Collection Name	Key Fields Stored	Description of Stored Data	Security Implication
sessions	sessionID (Random UUID), counselorID (Admin link), startTime, messages (Array)	Stores metadata about the session and the array of all chat messages.	Anonymous Link: All data is tied to the temporary sessionID, which is untraceable to a real person.
messages (Embedded in sessions)	Timestamp, senderType, cipherText	The actual, encrypted content of the communication sent by either the user or the counselor.	Zero-Knowledge Principle: The cipherText is the encrypted message. The server stores this unreadable text but does not hold the decryption key.

Collection Name	Key Fields Stored	Description of Stored Data	Security Implication
resources	keyword, link, description	Static or dynamic links to safety guides, helplines, and external support websites.	Supports the AI-Based Resource Guidance module.
stories	anonymousPostID, cipherText, moderationStatus	Encrypted content of stories shared by survivors, awaiting or having passed moderation.	Ensures that shared experiences are also protected by encryption.


Safety Exit / Panic Button Sequence Diagram
autonumber
actor =User= Contactor
=Client Application Frentend (React)
=Node.js/Express Server
=Node.js/Express Server API & Auth)
Local Trigger & Data Wipe
activate
C Clicks "Safety Exit" Button
Immediate UI feedback
Note right imedidemal data display, loading spinner)
Session Discomect
activate C
Local State Reset:Clear
C in-memory secrets
activate
Data Wipe: Clear Local Storage, Erases cathed messages, auth tokens, and user profile data
Request to invaiddatel data token
Session Disconect
activate C
POST //api/sessions/disconect
activate
C Token Invaliation: RevcJWT/Session ID Terminates all other logged-in sessions
Request to invalaate to tolens
activate
200 OK / Session Invalidated
Final Redirection
activate C
POST //api/session/disconted
Decoy Website
C Safe Redrect: "window.location replace(")
eag, Weather App, Search Engine
activate
D "Decoy Website

CHAPTER 8: CONCLUSION AND FUTURE WORK

8.1 Conclusion

HearMe successfully conceptualizes and defines the technical requirements for a secure, real-time crisis reporting platform. By integrating the high-performance MERN stack with mandatory application-layer End-to-End Encryption, the project fulfills the primary objective of establishing Zero-Knowledge Security. This architecture ensures that confidentiality is cryptographically guaranteed, directly addressing the trust deficit that prevents vulnerable populations from seeking help.[3] The rigorous definition of Non-Functional Requirements, focusing on performance, availability, and strict data minimization [7], demonstrates a comprehensive understanding of the high ethical and security standards required in health informatics and sensitive data management. HearMe stands as a technically proficient model for confidential digital support, demonstrating mastery over real-time communication systems and advanced cryptographic principles.

8.2 Future Work

To further enhance the HearMe platform's utility and technological sophistication, the following future development phases are proposed:

Phase II: Predictive Triage Integration

Future work should focus on integrating Artificial Intelligence (AI) exclusively for non-conversational diagnostic support. This module would utilize Machine Learning (ML) for symptom classification and risk prediction.[10] Crucially, this AI module must operate strictly on data that has been ethically reviewed and decrypted only by the authenticated counselor, or on non-sensitive session metadata, ensuring the integrity of the E2EE channel is never compromised. Research must prioritize the implementation of transparent and Explainable AI (XAI) to ensure clinical relevance and maintain the trust of professional counselors.[10]

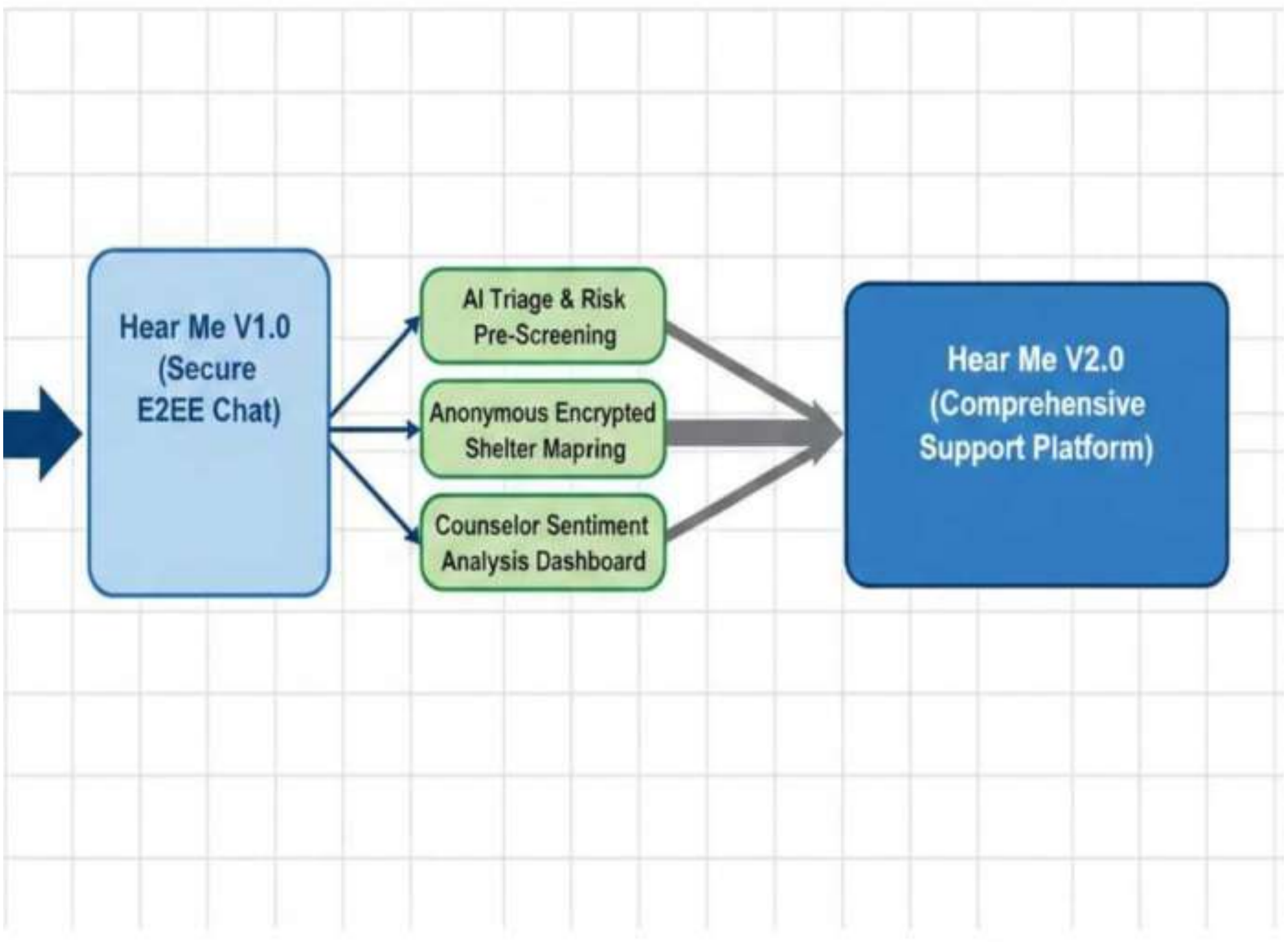
Phase III: Enhanced Real-time Protocols for Multimedia

To support richer forms of crisis intervention, such as Voice over IP (VoIP) or video chat, the system's communication protocols must be advanced. This includes exploring the migration or integration of Datagram Transport Layer Security (DTLS). DTLS is derived from TLS but is designed for connectionless, unreliable datagrams (UDP), which offers optimized performance for low-latency media applications while maintaining the integrity and privacy standards derived from SSL/TLS.[12]

Phase IV: Decentralization and Web 3.0 Exploration

To achieve the ultimate degree of user anonymity and data sovereignty, future work may involve exploring architectural migration pathways to decentralized systems or Web 3.0 frameworks. Utilizing decentralized identity mechanisms could further de-link session identifiers from any centralized organizational authority. The burgeoning mental health space within Web 3.0 and the Metaverse presents impressive potential for expanding access and anonymous interaction models.[19]

Hear Me Future Scope (V2.0 Expansion)


Hear Me V1.0 (Secure E2EE Chat)
AI Triage & Risk Pre-Screening
Anonymous Encrypted Shelter Mapring
Counselor Sentiment Analysis Dashboard
Hear Me V2.0 (Comprehensive Support Platform)

REFERENCES

1. http: //www .w3school .com/html/ 4
2. http: //www .w3school .com/css/ 4
3. https://bootstrap.com/ 4
4. https://Wikipedia.com/ 4
5. https://www.bmj .com/content/391/bmj .r2483 1
6. https://acf.gov/sites/default/files/documents/opre/advhocat frd report to opre 111918 508 compliant.pdf 5
7. https://pmc.ncbi.nlm.nih.gov/articles/PMC12623648/ 10
8. https://www.mentalhealthiournal.org/articles/minds-in-crisis-how-the-ai-revolution-is-impacting-mental-health.html 11
9. https://www.securitv.uci.edu/how-to/encrvption/ 13
10. https://www.checkpoint.com/cvber-hub/network-securitv/what-is-network-securitv/6-tvpes-of-network-securitv-protocols/ 12
11. https://www.bnxt.ai/blog/best-tech-stack-for-web-application-development-in-2025 2
12. https://dev.to/dimeloper/choosing-tech-stack-in-2025-a-practical-guide-4gll 15
13. https://plausible.io/securitv 7
14. https://wemedhealth.com/the-ultimate-guide-to-data-protection-in-health-apps/ 14
15. https://www.ibm.com/think/topics/end-to-end-encrvption 3
16. https: //www.egnvte.com/file -server/securitv-privacv 17
17. https://stackoverflow.com/questions/16475979/what-is-the-difference-between-functional-and-non-functional-
• 18 requirements
18. https://www.requiment.com/what-are-functional-and-non-functional-requirements/ 8
19. https://acquaintsoff.com/blog/mern-stack-app-deplovment-guide 9
20. https://www.geeksforgeeks.org/mern/mern-stack-proiect-deplovment-a-step-bv-step-guide/ 16
21. https://oiphi.imir.org/2025/1/e78065 6
22. https://gpsvch.bmj .com/content/35/4/e100825 19

APPENDIX -1: GLOSSARY OF TERMS

SDLC (Software Development Life Cycle): A systematic process used by the software industry to structure, plan, create, test, and deploy an information system.[4]

E2EE (End-to-End Encryption): A secure communication method that encrypts data on the sender's device and keeps it encrypted during transmission, decrypting it only on the recipient's device, ensuring that service providers cannot access the plaintext content.[3]

MERN Stack: An acronym for the MongoDB, Express.js, React.js, and Node.js technologies, forming a powerful, scalable technology stack for building modern web applications, particularly well-suited for realtime systems.[2]

TLS (Transport Layer Security): A cryptographic protocol designed to provide communication security over a computer network, widely used in web browsing (HTTPS) to ensure data encryption in transit.[13]

HTTPS (Hypertext Transfer Protocol Secure): The secure version of HTTP, implemented by running the HTTP protocol within an SSL/TLS wrapper to provide data encryption, integrity protection, and authentication between a client's browser and a web server.[12]

PHI (Protected Health Information): Any individually identifiable health information created or received by a covered entity, requiring stringent data protection measures, including encryption and access controls.[14]

Data Minimization: A core privacy principle requiring developers to collect and store only the minimum amount of personal or sensitive data necessary to provide the application's services, thereby minimizing the risk exposure in case of a breach.[7]

NFR (Non-Functional Requirement): Requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors, typically covering areas such as security, performance, scalability, and maintainability.[18]

Asynchronous Architecture: A system design where processes run independently and do not block the execution of subsequent tasks, allowing a system (like Node.js) to manage a large volume of concurrent connections efficiently, necessary for real-time chat.[2]

DTLS (Datagram Transport Layer Security): A protocol derived from SSL/TLS but designed for use with connectionless, unreliable datagrams (UDP), ensuring data integrity and privacy for low-latency applications like VoIP and video conferencing.[12]

PII (Personally Identifiable Information): Information that can be used to directly or indirectly identify an individual. In high-stakes crisis platforms, the collection of PII is deliberately avoided to protect anonymity.

Works cited

1. Nearly a billion women face domestic or sexual violence, report finds | The BMJ, accessed on November 28, 2025, https://www.bmj .com/content/391/bmj .r2483
2. Best Tech Stack for Web Application Development in 2025 - BNXT.ai, accessed on November 28, 2025, https://www.bnxt.ai/blog/best-tech-stack-for-web-application-development-in-2025
3. What is end-to-end encryption (E2EE)? - IBM, accessed on November 28, 2025,

https://www.ibm.com/think/topics/end-to-end-encryption

4. Minor Project-I Report Format (1).pdf
5. An Evaluation of the of the National Domestic Violence Hotline and loveisrespect - The Administration for Children and Families, accessed on November 28, 2025, https://acf.gov/sites/default/files/documents/opre/advhocat frd report to opre 111918 508 complia nt.pdf
6. Technology for Mental Health: Reflections on Scope and Future Directions in Institutes of Higher Education in India, accessed on November 28, 2025, https://oiphi.imir.org/2025/1/e78065
7. Plausible Analytics Security Practices, accessed on November 28, 2025, https://plausible.io/security
8. What are Functional and Non-functional Requirements?, accessed on November 28, 2025, https://www.requiment.com/what-are-functional-and-non-functional-requirements/
9. Deploying MERN Stack Apps: Cloud, Serverless, and Best Practices, accessed on November 28, 2025, https://acquaintsoft.com/blog/mern-stack-app-deployment-guide
10. Artificial intelligence for mental health: A narrative review of applications, challenges, and future directions in digital health - PubMed Central, accessed on November 28, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC12623648/
11. Minds in Crisis: How the AI Revolution is Impacting Mental Health, accessed on November 28, 2025, https://www.mentalhealthiournal.org/articles/minds-in-crisis-how-the-ai-revolution-is-impacting-mental-health.html
12. 6 Types of Network Security Protocols - Check Point Software, accessed on November 28, 2025, https://www.checkpoint.com/cyber-hub/network-security/what-is-network-security/6-types-of-network-security-protocols/
13. Protect Data with Encryption - UCI Information Security - UC Irvine, accessed on November 28, 2025, https://www.security.uci. edu/how-to/encryption/
14. The Ultimate Guide to Data Protection in Health Apps - weMED Clinics, accessed on November 28, 2025, https://wemedhealth.com/the-ultimate-guide-to-data-protection-in-health-apps/
15. Choosing Tech Stack in 2025: A Practical Guide - DEV Community, accessed on November 28, 2025, https://dev.to/dimeloper/choosing-tech-stack-in-2025-a-practical-guide-4gll
16. MERN Stack Project Deployment - A Step-by-Step Guide - GeeksforGeeks, accessed on November 28, 2025, https://www.geeksforgeeks.org/mern/mern-stack-proiect-deplovment-a-step-bv-step-guide/
17. Security Architecture for End-to-End Data Protection - Egnyte, accessed on November 28, 2025, https://www.egnvte.com/flle-server/securitv-privacv
18. What is the difference between functional and non-functional requirements? [closed] - Stack Overflow, accessed on November 28, 2025, https://stackoverflow.com/ questions/16475979/what-is-the-difference-between-functional-and-non-functional-requirements
19. Future of mental health in the metaverse | General Psychiatry, accessed on November 28, 2025, https://gpsych.bmi.com/content/35/4/e100825