BSCS-S25-006

03-134221-039 SYEDA HIRA MEHBOOB

03-134221-034 SAMAHA MUNIR

Fluenti: AI Powered Speech Therapy & Emotional Support

In partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science

Supervisor: Dr Junaid Nasir Qureshi

Department of Computer Sciences

Bahria University, Lahore Campus

January 2026

C e r t i f i c a t e

We accept the work contained in the report titled

"Fluenti: AI Powered Speech Therapy & Emotional Support"

written by

SYEDA HIRA MEHBOOB

SAMAHA MUNIR

As confirmation of the required standard for the partial fulfilment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: Dr Junaid Nasir Qureshi

_____ (Signature)

January 05, 2026

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-134221-039	SYEDA HIRA MEHBOOB	
03-134221-034	SAMAHA MUNIR	

Date : January 05, 2026

Especially dedicated to
my beloved mother, father and brothers
(SYEDA HIRA MEHBOOB)
my beloved mother, husband and sister
(SAMAHA MUNIR)

ACKNOWLEDGEMENTS

We would like to thank everyone who contributed to the successful completion of this project. We would like to express our gratitude to our research supervisor, Mr. Junaid Nasir, for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, we would also like to express our gratitude to our loving parents and friends who have helped and given us encouragement

SYEDA HIRA MEHBOOB
SAMAHA MUNIR

Fluenti: AI Powered Speech Therapy & Emotional Support

ABSTRACT

The barriers to speech therapy and emotional support also include increased accessibility and inconsistent quality, relating to both children with speech disorders and adults who need mental health assistance. Such issues can be tackled through this project the creation of Fluenti, which is the union of interactive story building games that can be used in speech therapy with AI oriented emotional support sessions aimed at adults. The architecture uses a multi layered design where the frontend will be based on React and TypeScript and the backend will be based on Node.js and Express.js, with the specialized therapist services being based on Python Flask. In the case of speech therapy, Google Gemini creates age specific interactive narratives and carries out evaluations in four types of therapies, which are pronunciation, fluency, Developmental Language Disorder (DLD), and social communication. The story game uses Web Speech APIs browser native and speech recognition in real-time, dynamically changing difficulty level, and progress tracking, which is gamified. To offer emotional support, a Python-based therapy bot, which is driven by Groq LLM, offers responses with hybrid crisis detection (AI and pattern-based) and psychological profiling and session management. The site combines OpenAI Whisper API with speech-to-text and OpenAI TTS with text-to-speech to allow the voice responses. MongoDB takes care of user progress, session history, and psychological profiles data persistence. The security features are JWT authentication, two-factor authentication, encryption and rate limiting. The system is a connection between therapeutic interventions and modern technology, which offers accessible, scaled, and individual support. The further development will focus on a broader range of types of therapies, improved crisis-detecting algorithms, and the system modification to multilingual support and global markets.

TABLE OF CONTENTS

Contents

DECLARATION 2
ACKNOWLEDGEMENTS 4
ABSTRACT 5
TABLE OF CONTENTS 6
LIST OF TABLES 9
LIST OF FIGURES 10
LIST OF APPENDICES 12

CHAPTER 1 13

INTRODUCTION 13

1.1 Background 13
1.2 Problem Statements 14
1.3 Aims and Objectives 16
1.4 Scope of Project 17

CHAPTER 2 18

LITERATURE REVIEW 18

2.1 AI-Powered Speech Therapy for Children 18
2.1.1 Gamification in Speech Therapy 19
2.1.2 AI-Based Assessment 19
2.1.3 Story-Based Learning Systems 20
2.2 AI-Driven Emotional Support Systems 20
2.2.1 Conversational AI for Mental Health 20
2.2.2 Crisis Detection and Intervention 21
2.2.3 Psychological Profiling and Analysis 21

CHAPTER 3 22

DESIGN AND METHODOLOGY 22

3.1 System Architecture Overview 22
3.1.1 Frontend Architecture Design 23
3.1.2 Backend Architecture Design 25
3.1.3 AI/ML Services Integration 26
3.1.4 Database Architecture 28
3.1.5 System Use Case 29
3.2 Data Management and Storage 29
3.2.1 Database Schema Design 30
3.2.2 Data Collection and Processing 31
3.2.3 Data Flow Architecture 32
3.3 AI-Powered Story Game for Speech Therapy 34
3.3.1 Story Generation System Design 35
3.3.2 Assessment Methodology 36

3.3.3 Speech Recognition and Processing....37
3.3.4 Game Logic and Progress Tracking....37
3.4 AI-Driven Emotional Support System 38
3.4.1 Conversational Therapy Bot Design....40
3.4.2 Knowledge Base and RAG Implementation....41
3.4.3 Crisis Detection Methodology....42
3.4.4 Psychological Profiling System....43
3.5 Speech Processing Integration....45
3.6 Authentication and Security Design....47
3.6.1 JWT Authentication System....47
3.6.2 Two-Factor Authentication Implementation....47
3.6.3 Data Encryption and Protection....47
3.6.4 Rate Limiting and Audit Logging....48
3.7 User Interface Design....48
3.7.1 Adult Dashboard Design....48
3.7.2 Emotional Support Interface Design....49
3.7.3 Child Dashboard Design....50
3.7.4 Story Game Interface Design....50
3.7.5 Landing Page and Therapist Finder Design....51
3.8 Implementation Methodology....52
3.8.1 Development Environment Setup....52
3.8.2 Service Deployment Architecture....53
3.8.3 Error Handling and Retry Mechanisms....53

CHAPTER 4....55

SYSTEM IMPLEMENTATION....55

4.1 Development Environment Setup....55
4.1.1 Technology Stack and Tools....55
4.2 Backend Implementation....56
4.2.1 Node.js API Gateway Logic....56
4.2.2 Python Flask Therapy Service....58
4.2.3 Hybrid Crisis Detection Algorithm....59
4.3 Frontend Implementation....59
4.3.1 Story Game Logic....59
4.3.2 Voice Integration Logic....61
4.4 Database & Security Implementation....61
4.4.1 Mongoose Schemas....62
4.4.2 Data Encryption....62
4.5 Deployment....62

CHAPTER 5....63

RESULTS AND EVALUATION....63

5.1 User Interface Results....63
5.1.1 Authentication & Onboarding....63
5.1.2 Child Dashboard & Story Game....64
5.1.3 Adult Dashboard & Therapy....64
5.2 Functional Testing Results....64

5.2.1 Authentication and Security Tests 65
5.2.2 Story Game Logic Tests (Child User) 65
5.2.3 Emotional Support & Crisis Tests (Adult User) 66
5.2.4 Data Persistence Tests 66
5.3 Performance Evaluation 67
5.3.1 AI Model Latency 67
5.3.2 Audio Processing Latency 67
5.4 Discussion of Constraints 68
5.5 Conclusion of Evaluation 68

CHAPTER 6 69

CONCLUSION AND RECOMMENDATIONS 69

6.1 Conclusion 69
6.2 Future Work and Recommendations 70

REFERENCES 71

APPENDICES 73

APPENDIX A: CODES 74

LIST OF TABLES

TABLE	TITLE	PAGE
Table 3.1	Frontend Technology Stack	24
Table 3.2	Backend API Routes Summary	26
Table 3.3	AI/ML Services Configuration	27
Table 3.4	Data Storage Characteristics	28
Table 3.5	Primary Database Collections and Purposes	30
Table 3.6	Therapy Types and Assessment Characteristics	36
Table 3.7	LLM Model Comparison for Emotional Support System	40
Table 3.8	Crisis Detection Levels and Harm Types	42
Table 3.9	Psychological Profile Components	44
Table 5.1	Security Tests	65
Table 5.2	Story Game Logic Tests	65
Table 5.3	Crisis Tests	66
Table 5.4	Data Persistence Tests	66
Table 5.5	Performance Evaluation	67

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 3.1	System Architecture Diagram	23
Figure 3.2	Frontend Architecture Diagram	24
Figure 3.3	Backend Architecture Diagram	25
Figure 3.4	AI/ML Services Integration Diagram	27
Figure 3.5	Database Schema Relationship ERD	28
Figure 3.6	Fluenti System Use Case Diagram	29
Figure 3.7	Data Collection and Processing Flow	32
Figure 3.8	Data Flow Architecture Diagram	34
Figure 3.9	Sequence Diagram for Story Game System and Flow	38
Figure 3.10	Sequence Diagram for Emotional Support System	41
Figure 3.11	Crisis Detection Hybrid Methodology	43
Figure 3.12	Speech Processing Integration UML Diagram	46
Figure 3.13	Emotional/Mental Therapy Dashboard	49

Figure 3.14	Emotional/Mental Therapy Chat Mode	49
Figure 3.15	Speech Therapy Dashboard	50
Figure 3.16	Story Game Interface	51
Figure 3.17	Home/Landing Page	52
Figure 3.18	Therapist Finder Design on Landing Page	52
Figure 4.1	Request Handling Flowchart	57
Figure 4.2	Game State Machine Diagram	60
Figure A.1	Pattern-Based Crisis Detection Engine	74
Figure A.2	AI-Powered Crisis Analysis Prompt	74
Figure A.3	Hybrid Reconciliation Logic	75
Figure A.4	Session Isolation and Memory Management	75
Figure A.5	Assessment Prompt Engineering	76
Figure A.6	Story Game Logic & Challenge Timing	76
Figure A.7	Structured JSON Output Schema	77
Figure A.8	API Retry Mechanism & Error Handling	77

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
APPENDIX A: Codes		70

CHAPTER 1

INTRODUCTION

1.1 Background

The rates of speech / language disorder among the pediatric population are high, and studies show that Speech and language developmental delays in children can be estimated by age and measurement, yet the prevalence estimates differ considerably, with many studies estimating prevalence as around 5-12% [1]. On the same note, there are a significant number of adults worldwide faced with mental health challenges, and The World Health Organization states that as of 2021, almost 1 out of 7 individuals (an estimated 1.1 billion) develop a mental disorder, and the most prevalent ones are anxiety and depressive disorders. [2]. Although these conditions are common, they do not receive quality therapeutic services because of geographical barriers, lack of professional therapists and of long waiting lists and poor quality of service delivery in different regions.

Conventional speech therapy in children normally involves frequent face-to-face sessions with certified speech-language pathologists, which may not be easy to the families because of time factors, transportation challenges, and availability of experts especially in underserved regions. Some of the most mentioned factors by families and adults as impediments to receiving high-quality therapy include geographic distance, cost, lack of clinician availability, and long wait lists; telepractice solves some of the issues related to accessing high-quality therapy but presents its own equity concerns (internet, device access). [3]. On the same note, the adults who require emotional support and mental health services are also likely to be hindered by such factors as stigma, a lack of therapists, incompatibility of schedules, and irregularity in the quality of the services offered.

Digital technologies and artificial intelligence are promising to decrease access barriers because they will allow automated assessment, remote interventions, and individual content. Despite its encouraging outcomes, a large number of tools are at the stage of prototype or pilot-study, and large-scale trials are still uncommon. [4]. The AI-based systems may give individualized treatment regimes, undertake examination, and offer a uniform quality of care to the patients irrespective of their geographical setting. Conversation agents powered by AI have been demonstrated to be effective in offering mental health assistance, with 24/7

accessibility and lowering the obstacle to emotional support services. Conversational agents (chatbots) demonstrate the short-term efficiency in symptoms of depression and anxiety reduction, well-being and distress decrease, but the long-term efficiency is unproven, and systematic reviews indicate safety, ethical, and quality-control risks - the use of them is being formulated [5][6]. On the same note, Gamification has been mentioned as one of the effective ways of assisting children to engage in therapeutic processes, and studies have revealed that the game usage can positively affect motivation, adherence, and treatment success in speech therapy in children [7].

This study will be focused on creating Fluenti, a complex AI-based application that will help solve the issue of accessibility in both speech therapy and emotional support, by integrating interactive story-building games in pediatric speech therapy with the application of emotion support sessions driven by AI, in adults.

1.2 Problem Statements

The speech therapy and emotional support services have a challenge due to the lack of accessibility as well as reliability when it comes to children with speech disorders and adults who are in need of mental health support. Intervention therapies must be dependable so that they could have the most favorable results and such issues as the lack of therapists, the overwhelming numbers of patients on the waiting list, and unreliable evaluation processes are not reliable inefficiencies that require to be corrected.

The impossibility to access the speech therapy services disrupts the balance in the requirements of the early intervention and the resources. Research indicates that about 5% to 12% of children develop communication issues to the extent they need to be treated [1]. However, families tend to miss some barriers which include geographical constraints, lack of therapists, time limitations, and a long waiting period before the families can get their services [3]. This could deny children the valuable opportunities of early interventions in case there is delay in services, and this could lead to long-term communication difficulties, academic difficulties, and social isolation. In case a service cannot be accessed due to cost or location the family can simply not undergo therapy which can only worsen the long term results. The extreme rates of such accessibility barriers make traditional therapy delivery quite ineffective and a functional system needs to be established that can be able to provide stable and readily available therapeutic support.

This problem is created by inconsistency of the same therapeutic interventions, which are provided by other providers and create confusion in the family and affect the outcome of the treatment. The quality of the speech therapy as well as the emotional support services could significantly vary, and the assessment techniques, as well as the approach to the intervention, could also significantly differ among the providers. These discrepancies occurring due to the variations in training, subjective assessment, and lack of a unified approach of assessment prove the need to have a standardized method, which would enable the provision of uniform and evidence-based therapeutic interventions across platforms. Similarly, the studies suggest that about one out of seven individuals in the world are living with mental illnesses [2], adults in need of mental health care are frequent subjects of unequal treatment in assessment and support, and there is evidence of variability in the efficacy and safety of conversational agents [6].

1.3 Aims and Objectives

The objectives of the thesis include:

i) To create one AI-based platform that combines interactive story-building games used in pediatric speech therapy with AI-based emotional support sessions, that should take into account the barriers to accessibility and offer an equal number of therapeutic interventions on these two levels.
ii) To suggest a powerful structure with the use of advanced AI models such as Google Gemini to evaluate conditions and generate stories, Groq LLM to provide therapeutic responses based on empathy, and real-time speech recognition technologies to provide individualized, adaptive therapeutic interventions with standard quality evaluation protocols.
iii) To generally develop more authenticity and consistency in the delivery of therapeutic services in digital healthcare applications, specifically by adopting evidence-based assessment practices, hybrid crisis detection mechanisms, and psychological profiling mechanisms that can guarantee reliable and effective interventions both per the speech therapy requirements and the emotional support requirements.

1.4 Scope of Project

The idea that this project is based on is the creation of a single AI-based platform in speech therapy and the emotional support services. The deliverables include:

- React with TypeScript as the frontend and Node.js with Express.js as the back-end API, and Python Flask as a special therapy service with an interactive interface to pediatric speech therapy and adult emotional support.
- A pediatric speech therapy story-building game system powered by Google Gemini AI models to generate and compare stories and outcomes in four types of therapy (pronunciation, fluency, Developmental Language Disorder, and social communication) with real-time speech recognition and adaptive level of difficulty and feedback with a game.
- A chat and voice-based emotional support system based on Groq LLM that can offer empathetic therapeutic feedback to users based on mental health datasets through Retrieval-Augmented Generation (RAG), an evidence-based therapeutic system, and integrate OpenAI Whisper API and OpenAI TTS.
- A MongoDB database system to store persisted data of user progress, session history, psychological profiles and therapy statistics to track and provide long term progress analysis.
- A system of authentication and security such as JWT based authentication, two factor authentication, data encryption, rate limiting and audit logging that will assure the safe access and protection of sensitive therapeutic data.

CHAPTER 2

LITERATURE REVIEW

The use of Artificial Intelligence (AI) has become one of the key tools in filling global gaps in the field of speech therapy and mental-health care. In the past 10 years, AI-based systems were implemented to complement conventional services, improve accessibility, automate assessment and offer interactive therapeutic experiences. Although it promises progress, it is still not implemented evenly with most of the current systems not being integrated, having no oversight, or decision frameworks based on evidence [8]. The literature always stresses that there should be available, personalized, and quality interventions, especially to the groups who have limited access to professional clinicians. AI-assisted speech therapy systems can help children with communication disorders, whereas AI-assisted emotional support systems can be used to address the surging demand of mental-health care services in underserved areas. Nevertheless, the existing technologies possess some issues like their inability to be accurate, irregular empathy modeling, poor contextual awareness, and lack of proper safety measures in crisis situations. This chapter summarizes significant areas of research in relation to speech therapy technologies, game, assessment, conversational agents, crisis detection, and psychological profiling.

2.1 AI-Powered Speech Therapy for Children

The AI-assisted speech therapy has developed considerable interest because of its ability to expand access, automate feedback, and supplement care. Speech-language pathologists depend on standardized assessment instruments, practice, and customized treatment programs; although most of the families struggle to obtain timely services due to geographic, financial, or human resource constraints [1][3]. The most recent developments in AI enable interactive therapy space, the analysis of articulation with the help of AI, and the adjustment of the difficulty level based on the development of the child. The scoping reviews feature the increasing ecosystem of mobile apps, web-based programs, and AI-enabled tools that assist with early intervention and offer organization of practice opportunities outside of the clinic context [4]. Typical of these systems are speech recognition, natural language processing as well as machine learning to provide real time feedback. All in all, studies indicate that, AI-enhanced speech therapy can be used as an addition to conventional approaches to provide flexible, engaging, and personalized therapeutic experiences, though further standardization and validation is still required.

2.1.1 Gamification in Speech Therapy

The use of gamification as a way of improving motivation, interest, and predictability in the speech therapy practice has been extensively examined. Systematic reviews emphasize that digital games, specifically those that encompass the elements of rewards, levels, and interactive narration, enhance compliance and make repetitive operations more entertaining to children [7]. Such systems frequently rely on animated character, progress tracking systems and real-time performance feedback to keep the user interested. Although these are the strengths, issues still exist. Existing literature is characterized with several limitation such as: Flawed speech-recognition models in disturbed conditions. Inappropriate correspondence between therapeutic goals and the mechanics of the game. Problem with adjusting the level of difficulty to personal speech skills. Few customizations regarding the various disorders. However, it is also shown that gamification provides a positive effect on motivation and practice frequency throughout the studies, and thus is an important element of the contemporary digital speech-therapy intervention [7].

2.1.2 AI-Based Assessment

Artificial intelligence (AI) models especially supervised learning, feature extraction, and NLP models have demonstrated the possibility of aiding evaluation of speech and language disorders. It has been proven that online speech-therapy services based on machine-learning classifiers can be used to help detect articulation disorders, fluency problems, and developmental communication challenges early [9]. The recent literature is also the one that investigates multimodal assessment in which acoustic features, rhythm, pauses, and syllable patterns are examined to identify certain conditions, such as stuttering. Indicatively, multimodal machine-learning methods of detecting stuttering show good results in the examination of speech timing characteristics and detection of disfluencies [10]. In spite of the fact that the AI tests are not meant to substitute an expert evaluation, they offer scalable screening methods that are able to assist clinicians and enhance early screening in underserved communities.

2.1.3 Story-Based Learning Systems

Therapeutic environments based on stories have been extensively employed to enhance language development in children such as vocabulary development, narrative structure, sequencing, and expressive communication. It has been found that narrative interventions have the advantage of improving engagement, situating speech practice and facilitating generalization of skills to academic and everyday environments [11]. Investigations conducted on children with Developmental Language Disorder (DLD) indicate that: Narrative retelling enhances the expressive language. Stories that focus on characters achieve motivational power. Developed storytelling favors the working memory and understanding. The interactive story contexts improve the word retrieval and sentence formulation. These results prove that story-based interventions as the useful and effective part of the speech-therapy program have a strong evidence.

2.2 AI-Driven Emotional Support Systems

The necessity of mental-health care around the globe has increased the pace at which AI-powered emotional assistance methods are adopted. The shortage of clinicians, geography, stigma, and prolonged wait time are some of the obstacles rendering AI-based assistance an attractive addition. Studies indicate that AI can deliver psychoeducation, mood monitoring and short-term emotional support, especially in the cases of mild to moderate psychological distress [5][12]. The tools of emotional support that are based on AI usually rely on large language models, rule-based reasoning, and sentiment analysis to interpret user input and respond to it. Nonetheless, this has been reviewed as saying that although these systems enhance access, they should be built to withstand ethical considerations, protocols in crisis-management and therapeutic frameworks that are based on evidence.

2.2.1 Conversational AI for Mental Health

The capacity to provide mental-health interventions has been researched with regard to conversational agents (CAs). The meta-analyses of the randomized controlled trials demonstrate that CA-based interventions may have a significant impact on the decrease of symptoms of depression and psychological distress in comparison with the control conditions [6]. It is the usual improvements in short-term interventions and they are normally varied based on the population and design of the system. Limitations are also pointed at in recent systematic reviews. Although contemporary LLMs are able to simulate empathetic conversational styles, they might not be predictable, emotionally sensitive, and context-sensitive in practice [8]. This contradiction lends credence to the fact that clear-cut

therapeutic models, customization measures, and human control are all required where necessary. In general, conversational AI is a potentially promising tool of emotional support that is available, though needs close monitoring and adaptation to evidence-based practices.

2.2.2 Crisis Detection and Intervention

A mental-health support system should have an essential element of crisis detection. The AI tools analyze texts, sentiment analysis, and time indicators and patterns of behavior to identify risk factors including self-harm thoughts or immediate emotional pain. Surveys of multimodal deep-learning systems show high accuracy, and some are capable of predicting arising risks better than human observers [13]. Nevertheless, research insists on the fact that: The difference in performance is great among datasets. Artificial intelligence is not sufficient to make emergency decisions. Accuracy is enhanced through the use of hybrid systems (AI + rule-based signals). There should be definite escalation lines and supervision on clinicians. The use of hybrid models of detection is always advised due to combination of linguistic cues and predetermined safety regulations as well as contextual indicators, the false positives are minimized, and uniformity of crisis-management procedures is enhanced.

2.2.3 Psychological Profiling and Analysis

Psychological profiling with AI assistance is the analysis of language features, behavior patterns, and interaction data and is used to deduce psychological indicators, including emotional state, patterns of stress, or cognitive predispositions. Systematic reviews indicate that machine-learning as well as deep-learning approaches are capable of detecting mental-health-specific linguistic-markers with moderate and strong precision and are conditioned on high-quality data [14]. The multimodal techniques which involve the integration of text, voice and behavioral characteristics are more insightful but more demanding of proper validation. It is emphasized in research that psychological profiling should be applied sparingly, morally, and in conjunction with examination and not as a diagnostic instrument on its own [15]. The sources have frequently claimed that AI can be effectively used in the emotional tracking, the mood detection, and the individual insights, provided that it is done responsibly and with adequate measures.

CHAPTER 3

DESIGN AND METHODOLOGY

The methodology describes the process of creating Fluenti, an AI-based speech therapy and emotional support application. It offers a step-by-step guide on the pathway between the system architecture and data management to the integration and deployment of an AI model. Dividing the methodology into several processes will guarantee a systematic approach to the situation when every element of the method leads to proper therapeutic interventions and individualized support. Each section of the methodology is imperative in the creation of system architecture design, AI model integration, data processing, security implementation and user interface development.

The methodology will consist of two primary sections: the former is dedicated to the AI-controlled system of a story game as the pediatric speech therapy approach and the latter is concerned with the AI-based system of the emotional support system in adults. All the components are subjected to architecture design, AI service integration, data management and implementation processes to make sure that the interventions of the speech therapy session and the emotional support session were implemented correctly to create a strong therapeutic platform. The systems are then tested based on performance, security and user experience and then deployed to real-life therapeutic use.

3.1 System Architecture Overview

The Fluenti system has been designed with a 3-tier architecture: frontend (React/TypeScript), backend (Node.js/Express), and a Python Ai-driven therapy Flask service. The frontend is user-interactive and the UI, the Node.js backend is API request-processing and authentication, and the Python service is therapy session with AI models. User data, sessions and progress are stored in MongoDB. There are external AI services (Google Gemini, Groq, OpenAI), which are embedded with APIs. The separation facilitates the scaling, maintenance and independent updates of service.

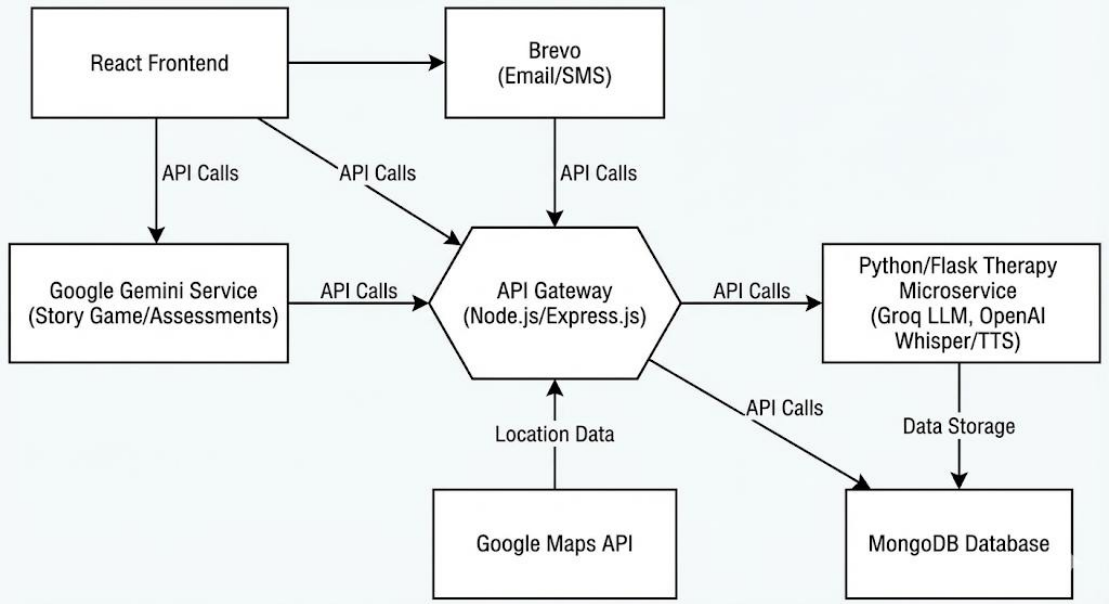


Figure 3.1 System Architecture Diagram

3.1.1 Frontend Architecture Design

The frontend code is written in React 18.3.1/ TypeScript. Wouter is dealing with client-side routing of public and protected routes. React Query deals with server status, caching and synchronization. UI also applies Radix UI and Tailwind CSS as a style, and Framer Motion as an animation. Vite is the build tool. The architecture has two types of users (children and adults) which have distinct dashboards and sets of features. Protected routes have authentication and user-type restrictions, so children can only get story game functionality and adults can only get emotional support functionality.

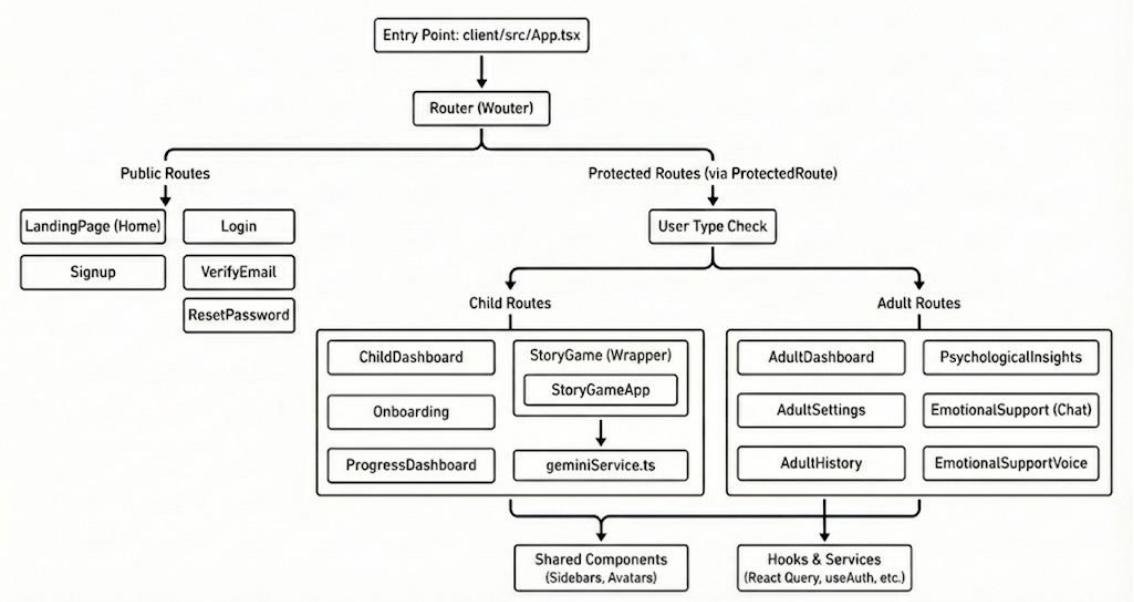


Figure 3.2 Frontend Architecture Diagram

Table 3.1: Frontend Technology Stack

Technology	Version/Purpose	Usage
React	18.3.1	Core UI framework
TypeScript	Latest	Type safety and development experience
Wouter	Latest	Client-side routing
React Query	Latest	Server state management and caching
Framer Motion	Latest	Animation library
Radix UI	Latest	Accessible component primitives
Tailwind CSS	Latest	Utility-first CSS framework
Vite	Latest	Build tool and dev server

3.1.2 Backend Architecture Design

The back-end is based on TypeScript, Express.js and Node.js. Express is used to handle HTTP requests, middleware and route registration. The architecture consists of authentication (JWT, 2FA), story game and emotional support API routes, the CORS middleware, the rate limiting middleware, and the JWT validation middleware, and the integration with MongoDB through the Mongoose. The backend receives therapy requests and sends them to the Python Flask service, and session persistence. Some of the security features are password hashing (bcrypt), token management, and audit logging.

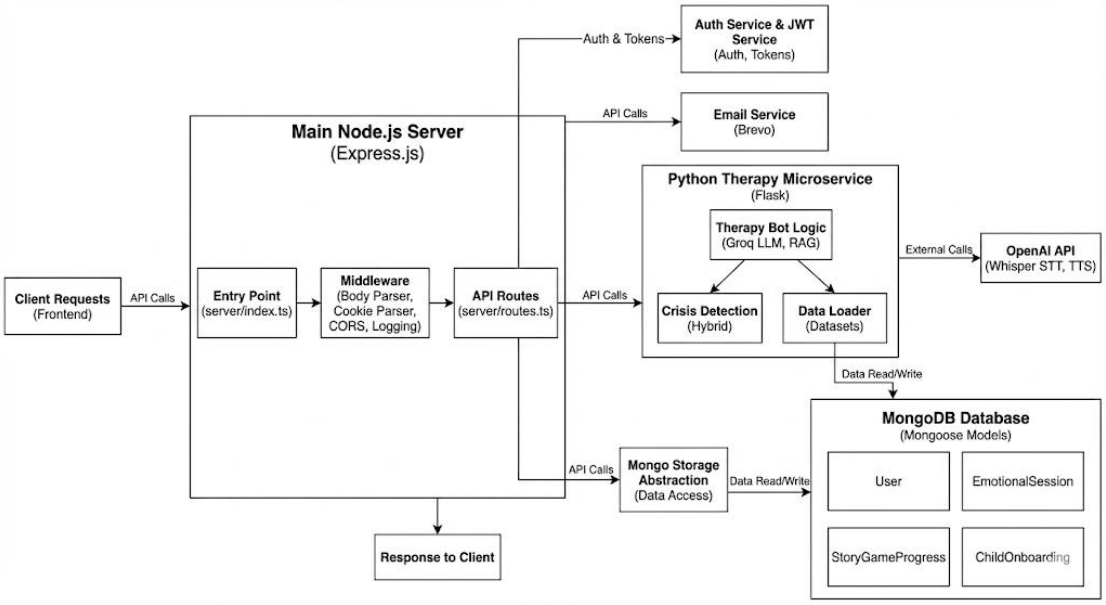


Figure 3.3 Backend Architecture Diagram

Table 3.2: Backend API Routes Summary

Route Category	Endpoint Prefix	Key Endpoints	Purpose
Authentication	/api/auth/	/login, /signup, /refresh, /verify-email	User authentication and authorization
Story Game	/api/story-game/	/progress, /session, /sessions	Story game data management
Emotional Support	/api/emotional-support	/chat, /emotional-support	Therapy session handling
Onboarding	/api/onboarding	/, /status	Child onboarding data
Psychological	/api/psychological-	/profile, /progress	Psychological insights
Therapist Finder	/api/therapists/	/find	Therapist location service

3.1.3 AI/ML Services Integration

The platform combines various AI and external services. Google Gemini is responsible in story generation, evaluation, and speech level analysis. The therapy bot is driven by Groq LLM with RAG to answer evidence-based questions. OpenAI Whisper API is used to do speech transcription and OpenAI TTS is used to do text to speak. Google Maps API allows therapist finder, which allows the user to find the nearest speech and mental health therapists, according to the geolocation. Brevo email service processes transactional emails such as verification emails, password reset, account lockout emails, and emergency crisis emails. To make these services reliable and perform well, they are combined through error management using REST APIs, and rate limiting with a recovery mechanism.

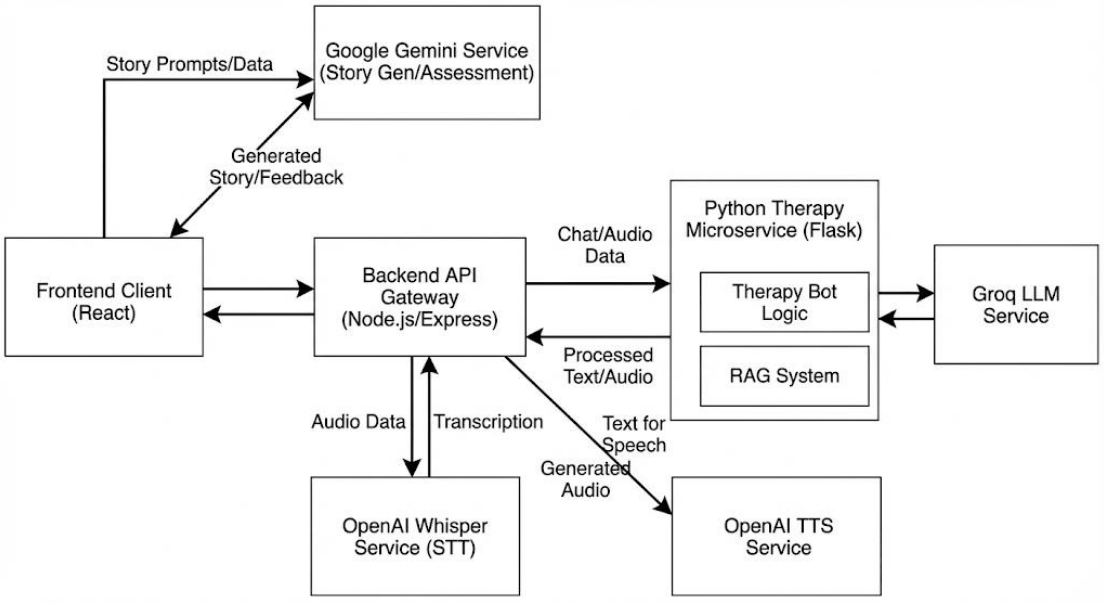


Figure 3.4 AI/ML Services Integration Diagram

Table 3.3: AI/ML Services Configuration

Service	Model/API	Primary Use Case	Integration Method
Google Gemini	2.5-flash	Story generation, story continuation	REST API via geminiService.ts
Google Gemini	2.5-pro	assessments, speech level evaluation	REST API via geminiService.ts
Groq	llama-3.3-70b-versatile	Therapy bot, empathetic responses	LangChain ChatGroq integration
OpenAI	Whisper API	Speech-to-text transcription	REST API via fastSTTService.ts
OpenAI	TTS API (tts-1)	Text-to-speech synthesis	REST API via enhancedTTSService .ts
OpenAI	text-embedding-3-small	Vector embeddings for RAG	LangChain OpenAI embeddings
Google Maps	Places API	Therapist location search	REST API via /api/therapists/find
Google Maps	Geocoding API	Coordinate accuracy	REST API via /api/therapists/find
Google Maps	Distance Matrix API	Road distance calculation	REST API via /api/therapists/find
Brevo	Email API	Email notifications, verification, crisis alerts	REST API via emailService.ts

3.1.4 Database Architecture

MongoDB contains user information, sessions, progress, psychological profiles. Mongoose is used to define schema and give validation. The collections are user, emotional session, story game progress, story game sessions, and child onboarding collections in the database. The data integrity, relationship and performance indexing are enforced using the schemas. Sensitive fields are encrypted and access of data is logged.

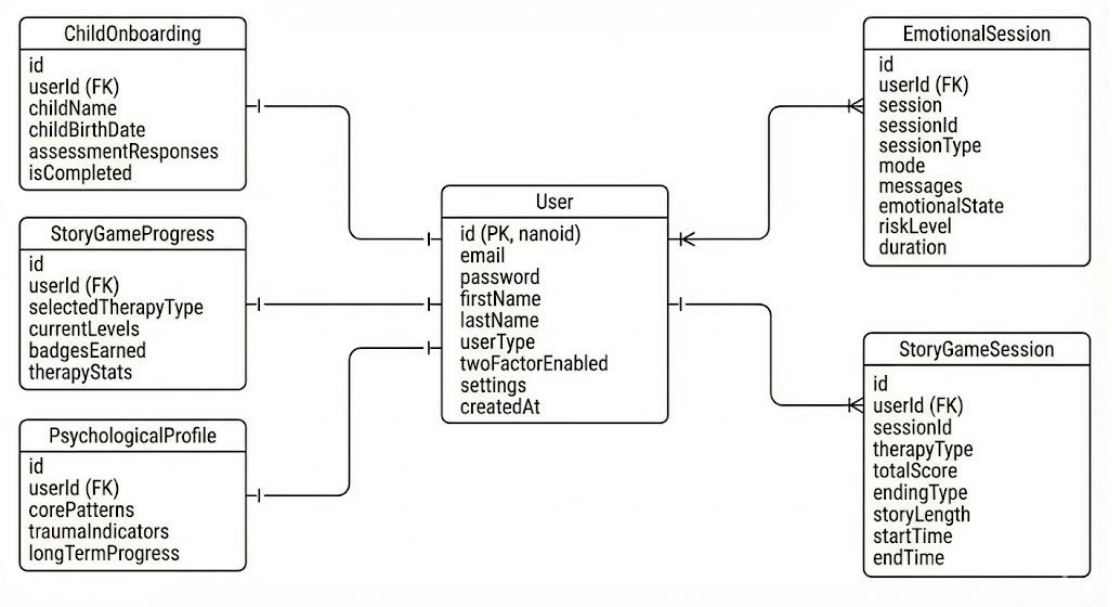


Figure 3.5: Database Schema Relationship ERD

Table 3.4: Data Storage Characteristics

Data Type	Storage Method	Encryption	Retention Period
User Credentials	Hashed (bcrypt)	Yes	Permanent
Conversation Data	Encrypted (Fernet)	Yes	30 days (configurable)
Session Data	Plain (non-sensitive)	No	Permanent
Progress Data	Plain (non-sensitive)	No	Permanent
Psychological Profiles	Encrypted	Yes	Permanent

3.1.5 System Use Case

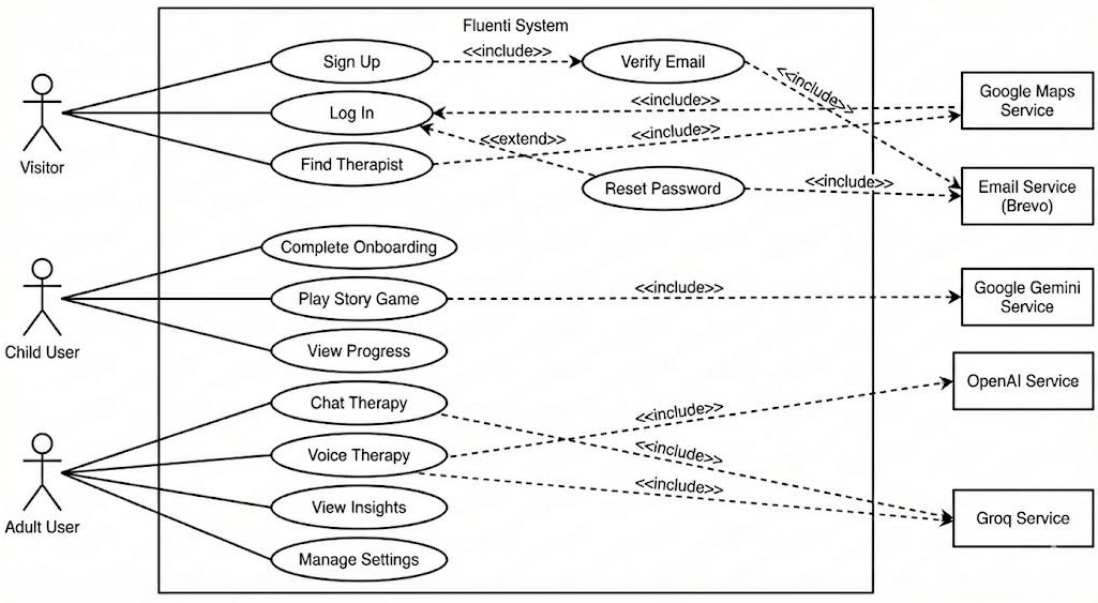


Figure 3.6: Fluenti System Use Case Diagram

Fluenti System Use Case Diagram This diagram outlines the functionality of the Fluenti platform, showing the interactions between three major actors (Visitor, Child User, Adult User) and their functional items. It shows some important integrations with the outside world, showing how the Story Game depends on the Google Gemini Service and how voice therapy integrates between OpenAI (speech) and Groq (intelligence). The workflows of secure authentication are also described in the diagram with a clear specification of the dependencies with Brevo Email Service where the user verification and password management take place. In general, it visualizes the system borders, along with the concrete third-party services to be taken to meet each user journey.

3.2 Data Management and Storage

MongoDB is the main database employed in the Fluenti platform, and the schema is designed based on child speech therapy as well as adult emotional support. There is data management that involves the collection, processing, encryption, and storage of data among various services. The system utilizes two databases MongoDB databases fluenti and therapysupportdb databases which are used to store Node.js service and Python therapy services respectively, respectively.

3.2.1 Database Schema Design

The database schema is a document based design that is used to optimize therapeutic data storage and retrieval. The main entities of the collections in MongoDB include users, therapy sessions, progress tracking, and psychological profiles. The schema design gives special attention to the referential integrity while using user ID references and facilitates the ability to query user specific data across several collections efficiently.

The core collections are User, EmotionalSession, StoryGameProgress, StoryGameSession and ChildOnboarding, with each of them having its purpose in the therapeutic workflow. User collection contains authentication credentials, security settings, and user preferences and therapy-specific collections contain session histories and progress measures. The schema contains hierarchical objects to represent complex data (e.g. assessment results, therapy statistics and psychological profiles) and this permits the structure of data to be stated in a flexible form without strict table constraints.

The indexing technique is based on commonly used fields like the fields of user id, email, session id, and timestamps so that retrieval of user sessions and progress data are fast. It is also designed with policies of data retention where the expired data is automatically deleted after 30 days, and the database can also support anonymization to ensure compliance with privacy. Collection-collection relationships are also ensured by relying on the option of userId references so that one can easily perform joins on the relationships and also data consistency within the platform.

Table 3.5: Primary Database Collections and Purposes

Collection Name	Primary Purpose	Key Relationships
User	Authentication, user profile, security settings	Referenced by all other collections via userId
EmotionalSession	Emotional support session history and messages	Linked to User, referenced by Python service
StoryGameProgress	Long-term progress tracking for speech therapy	Linked to User, updated per game session
StoryGameSession	Individual story game session records	Linked to User and StoryGameProgress

ChildOnboarding	Initial child information and assessment data	Linked to User, used for personalized therapy
Psychological Profiles	Long-term psychological insights and patterns	Linked to User, updated by Python therapy service
Crisis Logs	Crisis detection events and intervention records	Linked to User, used for safety monitoring

3.2.2 Data Collection and Processing

Collection of data is done at various entry points, which include the registration of the user, the play of story games, emotional support, and onboarding. In all data collection points, validation and sanitization are done in advance before the data is stored, which guarantees data integrity and security. The user input is also HTML escaped in addition to length and pattern checking, to avoid injection attack and assist with data quality requirements.

In the case of story games, the data of user speech is captured by use of browser-native Web Speech Recognition API and converted to text, which is then sent to the Gemini service to be processed further. The text that has been transcribed is validated to make sure that it is of at least a minimum quality, and game state information is gathered in steps as the game is played. The statistics of the progress are compiled during several sessions of the game, where the accuracy rates, the progression of the level, and the statistics of the challenges are calculated.

The data collection of emotional support is both text-based and voice-based, where the voice data needs to be transcribed using OpenAI Whisper API and then processed. The Python therapy service has elaborate input sanitization, such as HTML escaping, SQL injection pattern elimination, and length constraints of 5000 characters. Conversation data is encrypted with Fernet symmetric encryption prior to storage and user IDs are hashed to ensure that audit logs are not private.

Story game and emotional support systems have different data processing activities. The story game data are transmitted between the front and the Node.js backend to provide a communication with Google Gemini API to generate the stories and analyze the assessment. The data flow of emotional support is between frontend and Node.js backend and sent to the Python Flask service to process the requests of the therapy. The Python service does further processing such as crisis detection, psychological pattern analysis and knowledge base retrieval followed by the storage of encrypted conversation data.

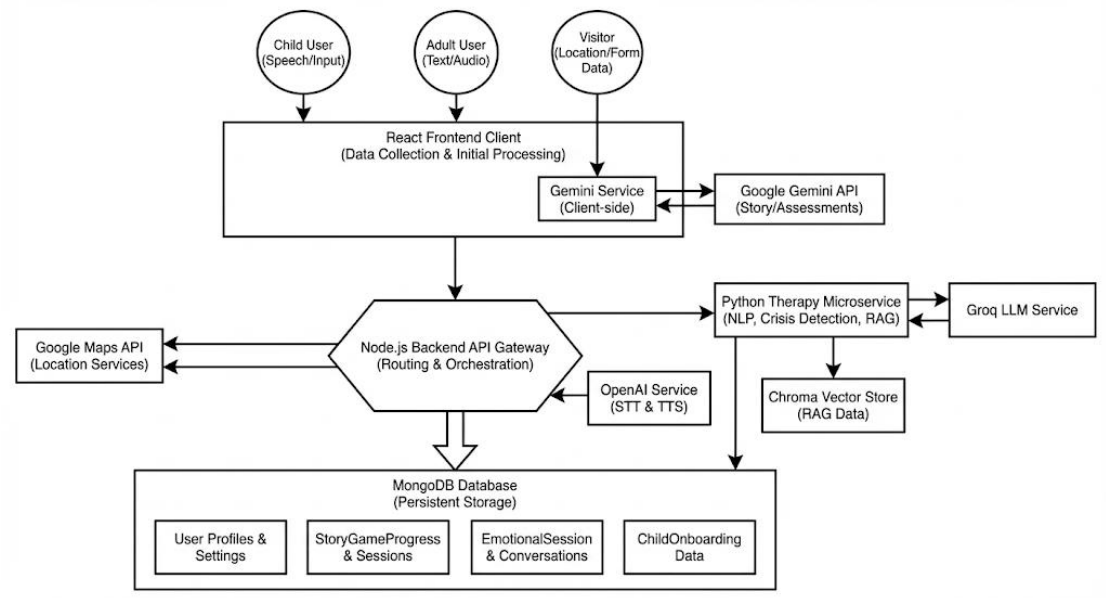


Figure 3.7: Data Collection and Processing Flow

3.2.3 Data Flow Architecture

The data flow architecture is organized in different directions between story game and emotional support features, and has common authentication and user management. The data in the story games moves in only one way, i.e. downstream to Gemini API, and the progress is saved to MongoDB on every major game event. The pipeline of emotional support data contains more elaborate components that are the Node.js backend, Python Flask service, and various AI services to provide transcription and generate therapy and speech-to-text services.

In the case of story game session, data flow is started once a user starts playing the game, and the game state is maintained locally in the browser in the localStorage to recover the game in the event of an interruption. Progress reports are submitted to the API on major milestones; the end of a game, completion of a level, and completion of an assessment. The backend is going to verify the data structure, modify the StoryGame Progress collection, and generate session records in the StoryGameSession collection. Story generation requests are generated in real time and sent directly to Google Gemini API bypassing the backend to optimize the performance, and only progress data is sent to the backend to be stored. The API will be based on the REST endpoints: GET /api/story-game/progress to get progress, POST /api/story-game/progress to save progress, and POST /api/story-game/session to save a session; the request and response data formats are in the form of JSON, and the authentication is provided by using JWT over the cookies of the httpOnly type.

Different stages are involved in the process of data processing of emotional support. In chat mode, the message sent by the user to the Nodes.js server is POST /api/emotional-support-chat where the message content and sessionid are sent. The backend sends the request to Python Flask service at /api/therapy/chat via HTTP POST request with the contents of the request as JSON. The Python service accesses the conversation history in MongoDB, detects crisis and psychological features and uses Groq LLM to create therapeutic response before sending it back with the metadata of the crisis and session. The Node.js server in its turn stores the whole conversation to the EmotionalSession collection. In voice mode, the flow consists of other steps: audio transcription using OpenAI Whisper API through POST /api/emotional-support with multipart/form-data, processing the text using the Python service, creating the response, and converting the text to speech using OpenAI TTS, and sending the audio back to the frontend, base64-encoded.

Session management architecture aids in creation of sessions and restoration of sessions. The Python service creates new sessions in case of no sessionId being presented and it provides secure session tokens (32-byte URL-safe tokens) expiring after 24 hours. Restoring the session Restoration of a session takes place when a request contains a sessionId, and the system retrievals data about the session in the EmotionalSession collection of MongoDB, decrypts conversation history and recovers session memory containing primary issues, progress notes, and conversation summaries. The decisions of session reuse are supported by AI, and the session can be continued automatically within 5 minutes by the user in case of returning. The isolation of the session is also strictly followed where each session has a separate memory to eliminate contamination across sessions. Error handling functions involve the use of retry in the case of service failure, graceful degradation in cases where services provided by the AI are not available, and full error logging to aid in debugging and monitoring.

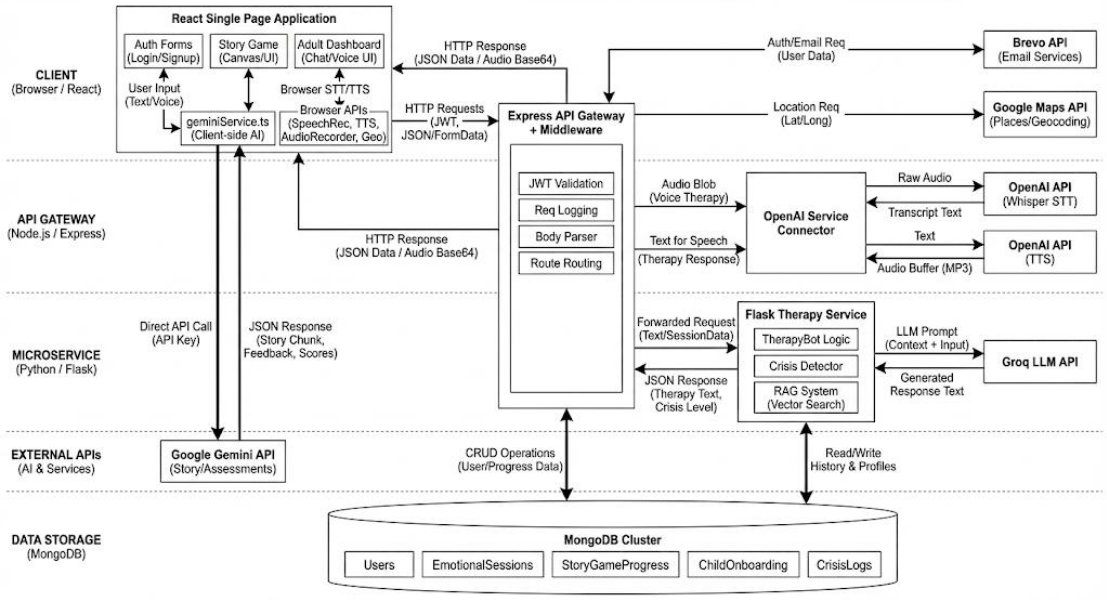


Figure 3.8: Data Flow Architecture Diagram

The data flow architecture guarantees the separation of concerns, where the Node.js backend will provide authentication, session management, and data persistence, and special purpose services (Python Flask to implement emotional therapy, Gemini API to generate stories) will do the domain-specific processing. The architecture is scalable and enables services to scale independently and provides consistency in data storage by having centralized MongoDB storage with appropriate indexing and relationship management. Services talk to each other and get standard HTTP with JSON messages, enabling interoperability and easy integration, and securing it wit`h JWT authentication and encrypted data storage as well as through verifying the security of a session token.

3.3 AI-Powered Story Game for Speech Therapy

The AI-driven story game will combine Google Gemini to generate stories and conduct assessments, speech recognition features that are browser-native and can be used in real-time, and adaptive control of the game to implement speech therapy individually. There are four types of therapy and the system facilitates them: pronunciation, fluency, Developmental Language Disorder (DLD) and social communication, each having a specific assessment process and a specific therapeutic intervention incorporated into a series of interactive stories.

Google Gemini was selected as it was used on story generation and evaluation rather than training a custom model due to the high level of language understanding and linguistic generation features without large training and massive computational capabilities. The pre-trained model also means that thousands of therapy-specific conversations do not need to be collected and curated and greatly decreases the cost and time of development. It is perfect to use in speech therapy because Gemini is able to produce content that is age-appropriate, comprehend the criteria of assessment, and modify the language complexity depending on the age of children. The real-time generation possibilities of the model allow starting the story immediately and proceeding with the assessment analysis, which is vital to keep the child actively engaged in the course of the therapy. Moreover, the constant model upgrading of Google allows access to the most recent language understanding advancements without training or model maintenance overhead. Gemini 2.5-flash, which is the system used to generate stories (faster response), and Gemini 2.5-pro, which is the system used to make evaluations (more accurate analysis), which relies on the understandings built in the model of language development, speech disorders and therapeutic interventions to support the therapy process with evidence-based therapy support.

3.3.1 Story Generation System Design

Story Generation is based on Google Gemini 2.5- flash, which is a real-time narrative generation prompting system, and the prompts are response to the age of the child, the type of the chosen therapy, and the level of ability of the child. The stories produced by the system are age-specific and have interesting themes and characters, and the therapeutic challenges are included in the narrative process. The story chunks are topped with open-ended questions and three creative action suggestions (2-4 words each) to help the child respond and preserve the narrative at the same time. The story generation process takes into account the age of the child based on the onboarding data, modifies the complexity of words, sentence structure and content of the stories. The ability to create their own stories, characters, settings, interests enables children to determine their own stories, characters and interests, and the AI provides context-sensitive suggestions at every step, allowing them to create their own adventures.

3.3.2 Assessment Methodology

Google Gemini 2.5-pro is used as a methodology of assessment of speech and language skills. The system performs pre-test evaluations prior to the commencement of gameplay whereby it produces therapy-specific prompts that become more difficult in three rounds. In the context of pronunciation therapy, the evaluation of articulation disorders can be conducted by repeating the target words in magic stories and evaluating substitutions, omissions, and distortions. Fluency tests involve progressively complex sentences which are aimed at stimulating stuttering behavior and are used to measure repetition, prolongation and blocks. The language delay that is tested by DLD is in the form of sentence complexity tasks that test the grammar, vocabulary and sentence structure. In social communication tests, one can be presented with the situation where perspective-taking, empathy, and social responsiveness are required. Each test is a text transcription of the speech of the child and the skill levels are rated 1-20 according to speech language pathology parameters, with the argument and child friendly rationale given about the result of each assessment.

Table 3.6: Therapy Types and Assessment Characteristics

Therapy Type	Focus	Assessment Method	Level Range	Key Evaluation Criteria
Pronunciation	Articulation disorders	Target word repetition in story context	1-20	Substitutions, omissions, distortions; sound accuracy
Fluency	Stuttering behaviors	Progressive sentence complexity	1-20	Repetitions, prolongations, blocks; speech flow
DLD	Language delay/disorder	Sentence complexity tasks	1-20	Grammar, vocabulary, sentence structure; linguistic complexity
Social Communication	Pragmatic communication	Scenario-based responses	1-20	Perspective-taking, empathy, social appropriateness

3.3.3 Speech Recognition and Processing

Speech processing and recognition makes use of Web Speech Recognition API which is browser-native and transcription of speech input of the child is real-time. The system is set to the recognition mode of continuous listening, English language in the system and interim results to enable immediate feedback. Silence detection mechanism is enabled by default to pause recognition after 2.5 seconds of silence and then the recognition restarts to allow natural pauses in speech. Visual feedback Visual feedback is pulsing orange rings, which appear around the microphone button during active listening, the character portrait animations are synchronized with the listening state. This transcribed text is then validated and trimmed and submitted to the Gemini service where they continue the story and are then subjected to therapeutic analysis. Text-to-speech narration employs the Web Speech Synthesis API of the browser, appropriate voices being selected automatically, high-quality natural voices having the priority, and a queued approach to speech ensures that there is no overlap of speech, and flow is continuous.

3.3.4 Game Logic and Progress Tracking

Game logic and progress tracking are a dual-mode system with differentiation between normal continuation of the story and evaluating a challenge. The application is controlled by the useReducer pattern of React to handle the state and store the state in local storage to resume in case of interruption. There are three measures of the scoring system: creativity score (0-10 per turn, zero during challenge), speech score (only during challenge: +5 to +15 success, -5 failure), and focus stars (0-3, +1 when you succeed in the challenge, and -1 when you fail). Challenge timing has strict rules: after a challenge has been made, then no further challenges may be made at all, two or more regular turns must be made before a challenge can be made. The system has two different modes: challenge evaluation mode which is the analysis of the reaction of a child on the challenge introduced previously and the regular story mode which continues the story and might introduce new challenges with references to the rules of timing. Win conditions are five challenges to finish the level to get a happy ending and a loss happens when the focus stars decrease to zero. The development statistics such as levels, assessments, badges and therapy statistics are stored in MongoDB upon major events in the game, allowing the long-term tracking of progress and customized therapy suggestions.

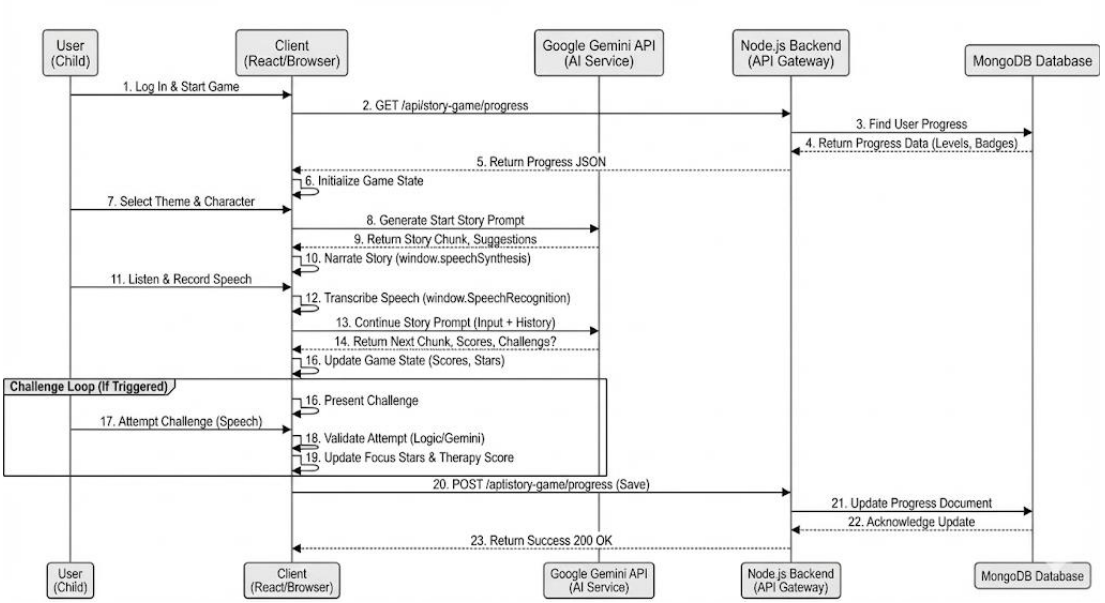


Figure 3.9: Sequence Diagram for Story Game System and Flow

There is a seamless integration of therapeutic interventions with entertaining stories through the story game system that also keeps kids motivated with gamification and also makes sure that the accuracy is maintained by using evidence-based assessment and intervention strategies. The adaptive difficulty system modulates the difficulty of challenge according to the level of the child at that time so that there is proper progression of the therapeutic process and avoids the frustration or boredom. The real-time feedback systems offer instant reinforcement to those who successfully complete the challenges and soft hints where to improve which facilitates the outcomes of therapy and engagement during the entire intervention process.

3.4 AI-Driven Emotional Support System

The emotional support system offers therapeutic chat service with Python-based service based on Groq LLM, a Retrieval-Augmented Generation (RAG) knowledge base, hybrid crisis detection, and psychological profiling. The system allows chat and voice, session control, encryption and evidence-based responses.

RAG has been selected over training a custom model since it is a combination of a pre-trained LLM and a knowledge base trained on mental health data, which allows providing evidence-based responses without the large training datasets or intensive fine-tuning. This can save the time and money of collecting and labeling the thousands of therapeutic conversations, making the development process much quicker and less infrastructure intensive. RAG enables the system to access therapeutic information that is part of the curated mental health datasets (such as counseling conversations, chatbot datasets, help-seeking conversations) and integrate it with the language understanding abilities of the LLM, in order to make sure that the responses are based on evidence-based practices. New research and therapeutic methods can be added to the knowledge base without retraining the model, allowing it to be flexible and in a state of constant improvement. Also, RAG makes it possible to retrieve contexts based on the psychological profile of the user, which allows the system to tailor responses to the user without losing the evidence-based therapeutic advice. It uses the conversational capabilities of the LLM with mental health data-specific knowledge to form a more trusted and sound therapeutic system than training a model on its own would provide.

The llama-3.3-70b-versatile of Groq was chosen to form part of the emotional support system since the inference engine offered by Groq is much faster than the standard cloud-based LLM services, which is crucial in the process of sustaining natural therapeutic conversations. The 70-billion parameter model provides excellent language understanding and generation which is applicable in therapeutic environments, with hardware acceleration capabilities provided by Groq to respond to queries in real-time with voice mode interaction requirements. The versatility of the model enables it to respond to different types of conversations when interacting with different people and even in crises by changing its response style accordingly. Also, the low price scheme of Groq allows conducting therapeutic dialogues in high volumes, and the API is reliable enough to remain available to customers who need to receive emotional support.

Table 3.7: LLM Model Comparison for Emotional Support System [16]

Model/Service	Speed	Cost	Context Window	Selection Reason
Groq llama-3.3-70b-versatile	Very Fast (Groq hardware)	Low	128K tokens	Selected - Fast inference, cost-effective, strong therapeutic capabilities
OpenAI GPT-4	Moderate	High	128K tokens	Too expensive for high-volume conversations
OpenAI GPT-3.5-turbo	Fast	Moderate	16K tokens	Good alternative but slower than Groq
Anthropic Claude 3	Moderate	High	200K tokens	High cost, slower inference
Meta Llama 3.1 / 3.3	Variable	Moderate	128K tokens	Requires infrastructure, maintenance overhead
Google Gemini Pro	Fast	Moderate	1M - 2M	Good option but Groq offers better speed
Mistral Large 2	Moderate	Moderate	128K tokens	Comparable but Groq's speed advantage

It is evident that the Groq llama-3.3-70b-versatile is the best choice in terms of speed, cost, and quality of the therapeutic conversation when used in the emotional support system with Groq hardware acceleration being the decisive factor when it comes to the quality of therapeutic interactions in real time.

3.4.1 Conversational Therapy Bot Design

The chatbot is a conversational type of therapy based on Groq and the ChatGroq model named lllama-3.3-70b-versatile. It has very strict session isolation, each session contains isolated memory tracking primary issues, progress notes, conversation summaries, key themes and user preferences. Session management encompasses the secure token authentication, decision-making on session reuse with the help of AI and contextual welcome information by the time of the day. An AI analysis of emotional depth, mental health topics, help seeking behavior and stage of conversation allows the robot to identify the type of response (casual, therapeutic or crisis). All answers are in a professional

therapeutic voice no matter how a user communicates with it, so they consistently offer a similar quality of the therapeutic process but can change according to the needs of the user.

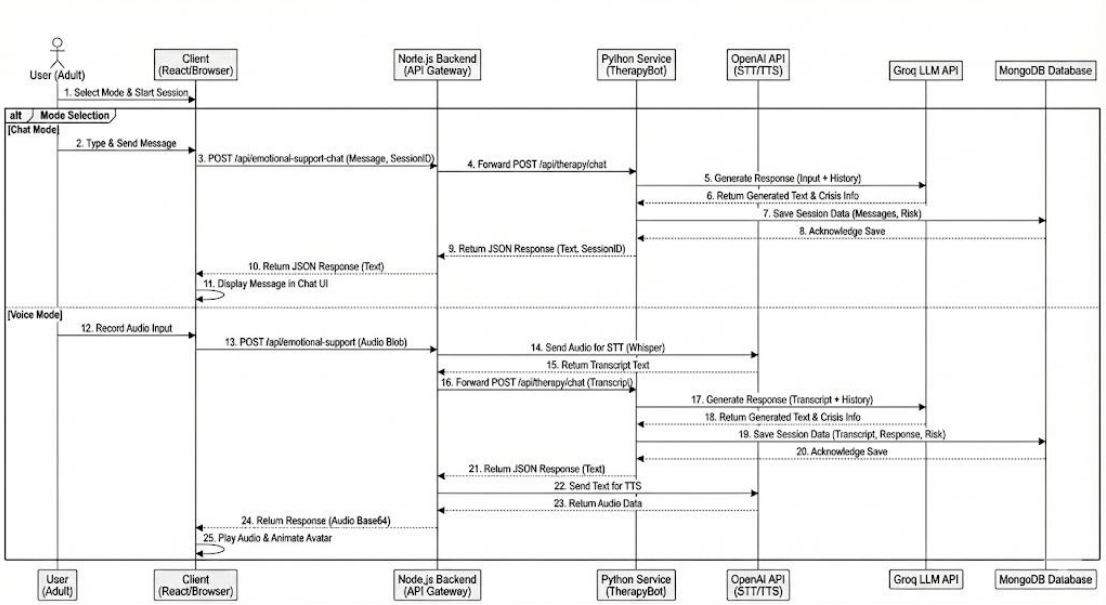


Figure 3.10: Sequence for Emotional Support System

3.4.2 Knowledge Base and RAG Implementation

The base of knowledge and RAG implementation process four data on mental health of Hugging Face: Amod/mentalhealthcounselingconversations (counseling conversations) [17], heliosbrahma/mentalhealthchatbotdataset (chatbot interactions) [18], nbertagnolli/counsel-chat (counseling chat data) [19], and nvidia/HelpSteer (help-seeking conversations) [20]. The streaming mode provides efficient loading, and the examples are processed on-the-fly, optimizing the use of memory, and, in case of streaming failure, it uses the traditional loading. Text processing starts with normalization via the normalizetext program that strips HTML/XML conventions, flattens quotations as well as straightening up whitespace. Quality test through the validatetextquality provides that the text should be of a standard of 20-2000 words on a minimum of 3 distinct words. The use of keyword matching in ismentalhealthrelevant is used to filter to find therapeutic content. Field extraction through extractdatasetfields is able to automatically deal with various dataset formats, extracting text in a variety of format including Context/Response pairs, question/answer pairs, and prompt/response pairs. The deduplication of the processed datasets is done to eliminate redundant entries. In text chunking, 1000 words in a chunk with 150 words overlap (15%) is made to ensure there is continuity of context in the chunk. The text, which has been processed and validated, is then transformed into the form of vector embeddings through the

text-embedding-3-small model of OpenAI, resulting in a Chroma vector store with metadata allowing similarity search to be done with ease. The system produces specialized retrievers, a crisis retriever (6 documents) to be used in crisis situations and a general retriever (6 documents) to be used in general therapeutic conversations. RAG retrieval is based on AI-enhanced queries to select and rank the most relevant context pieces, where word limits are 200 words in the situation of a casual interaction, 250 words in the situation of therapeutic response, and 300 words in the situation of crisis situation. The psychological profile gives knowledge base a boost, making it possible to retrieve profile-conscious contexts and provide personalized therapeutic response.

3.4.3 Crisis Detection Methodology

Table 3.8: Crisis Detection Levels and Harm Types

Crisis Level	Description	Response Action	Harm Type Detection
NONE	No crisis indicators detected	Standard therapeutic response	NONE
LOW	Minor emotional distress	Supportive response with resources	NONE
MEDIUM	Moderate distress requiring attention	Enhanced therapeutic response	NONE or SELF_HARM
HIGH	Significant risk requiring immediate support	Crisis intervention with helplines	SELF_HARM or HARM_TO_OTHERS
CRITICAL	Immediate danger requiring emergency response	Emergency notification + crisis resources	SELF_HARM, HARM_TO_OTHERS, or BOTH

Crisis detection involves a dual methodology of AI-based analysis and detection relying on patterns. The AI-based detection is based on the LLM analyzing text in crisis indicators, taking into account the context, intent, and emotional state and differentiating between general sadness and harm ideation. Pattern-based detection identifies linguistic features such as the word count, sentence count, the use of question marks, first and second person usage, use of negation words, intensity words, temporal words, and social words. It recognizes essential patterns, suicide ideation, self-harm, method, and immediate intent with the help of flexible regex patterns. The hybrid detector is smart enough to balance the differences between AI and pattern-based findings by incorporating AI analysis, which also operates with a safety-first mindset, according to which more risky tests should be performed. The

system has complex analysis to categorize interactions as minimal, simple, moderately, and complex to process heavy interactions faster. Things such as the detection of context analysis to avoid false positives (i.e. feeling down today versus want to die today), help-seeking behavior detection, which lowers crisis scores, the negation pattern that lowers crisis severity and time-of-day consideration that raises concern during vulnerable hours are all safety features. Harm type detection draws a difference between self-harm and harm to another person and ensures that safety measures prevent general emotional distress being incorrectly defined as self-harm.

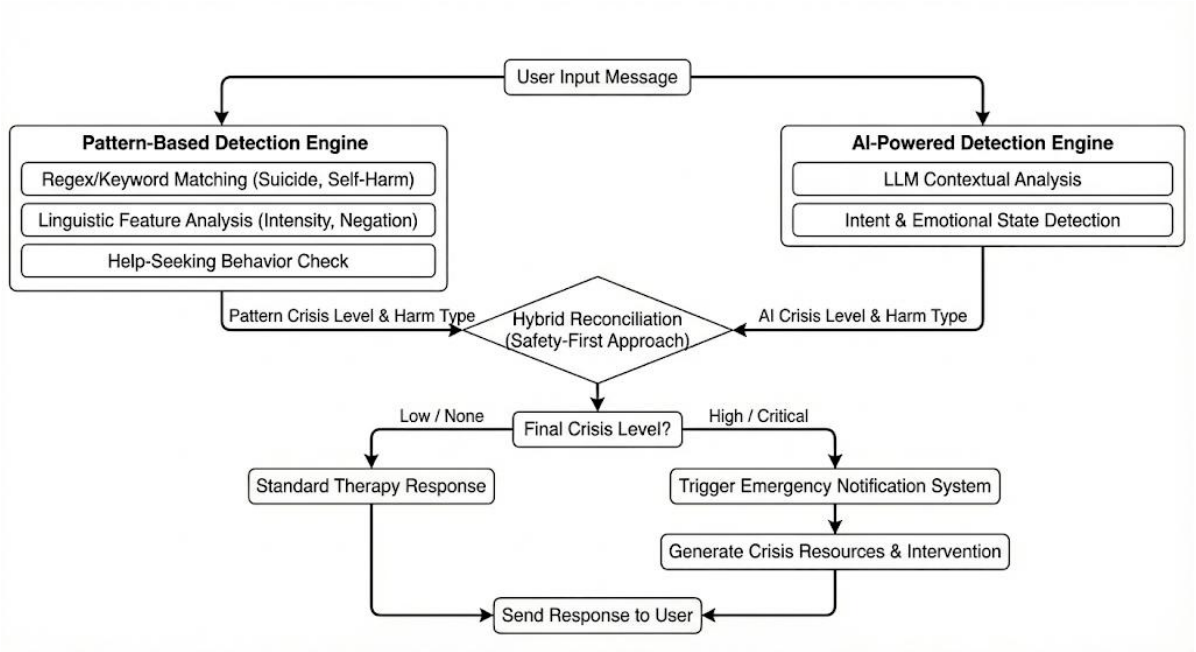


Figure 3.11: Crisis Detection Hybrid Methodology

3.4.4 Psychological Profiling System

The psychological profiling system works through analyzing patterns of conversation to create profiles, which are used to provide personalized therapeutic responses. The system operates with the help of the LLM-based analysis that helps to reveal the main behavioral patterns, cognitive patterns, emotion regulation patterns, and effective coping mechanisms. Indicating the presence of traumas is a sensitivity approach to AI analysis where trauma-related patterns are detected and the normal emotional responses are not over-detected. Long-term progress tracking tracks the progress over a period and assesses the risk trends by identification of improvements, worsenings, and no changes. The psychological profile involves the core patterns (recurring behavioral patterns), the trauma indicators (patterns when traumatic exist), cognitive patterns (thinking styles, processing patterns), emotional regulation patterns (how emotions are regulated), and coping mechanisms (effective

strategies found). After every conversation, profile updates are made and AI analysis draws new insights and incorporates them into the existing profile. Depending on the profile, the system creates tailored therapeutic advice, which allows profile-specific suggestions and adaptive intervention plans. Summary of the sessions also entails the psychological understanding based on the profile which gives the user long term developmental pointers and their individualized suggestions on what to further the growth.

Table 3.9: Psychological Profile Components

Component	Description	Update Frequency	Use Case
Core Patterns	Recurring behavioral themes	After each session	Personalized response generation
Trauma Indicators	Trauma-related patterns	When detected	Sensitive intervention strategies
Cognitive Patterns	Thinking styles and processing	After each session	Cognitive-behavioral interventions
Emotional Regulation	Emotion management patterns	After each session	Emotional support strategies
Coping Mechanisms	Effective coping strategies	When identified	Reinforcement and recommendations
Long-term Progress	Progress trends over time	After each session	Progress visualization and insights

The emotional support system combines security features such as Fernet symmetric encryption with PBKDF2 key derivation (100,000 iterations) of conversation sensitive data, input sanitization (HTML escaping and pattern removal) and per-user rate limiting (100 requests per hour) and secure audit logging without personally identifiable information. Session tokens are authenticated with expiry of 24 hours and all conversation information is encrypted and then stored in MongoDB. The system is chat and voice-enabled, and voice mode needs further enhancement with additional transcription with the help of OpenAI Whisper API and the transformation of the text to audio with the help of OpenAI TTS prior to audio responses are returned. The architecture provides evidence-driven therapeutic reactions with the help of RAG, customized interventions with the help of psychological profiling, and safety due to the thorough crisis detection, offering a safe and efficient platform to provide emotional support and mental health services.

3.5 Speech Processing Integration

Two methods are employed in speech processing, including browser-native APIs with the story game and cloud-based with emotional support voice mode. The story game is based on Web Speech Recognition API (speech-to-text) and Web Speech Synthesis API (text-to-speech) which are client-side technologies. The story game was selected to use Web Speech APIs as it allows interacting in real-time with a low latency, which is significant in the case of children engagement. They remove the overhead of backend processing and decrease the server load and API prices. Client-side architecture makes the architecture simpler, offline functionality is supported and shorter, simpler child speech is managed. Web Speech Recognition API is set with the continuous listening mode and English language option and interim results as a warning that it is working and will stop recognizing after 2.5 seconds otherwise there will be continuous listening. Web Speech Synthesis API involves automatic voice selection (with the highest priority on high-quality natural voices), a queue-based speech synthesis system (where a new synthesis task must finish before the same story segmentation can be repeated), and automatic narration (where a new piece of the story is generated and the system will automatically switch to narration), thus giving the effect of continuous narration without additional human intervention.

In the case of emotional support voice mode, the system runs OpenAI Whisper API on speech-to-text and OpenAI TTS on text-to-speech, both run server-side. Emotional support was selected as the openAI cloud services since they offer greater accuracy over longer, more complicated therapeutic dialogues, more emotional subtleties and speech patterns, and professional-grade TTS, which can be used in therapies. The server-side solution provides an opportunity to manage the session in a proper way, encrypt sensitive therapeutic data, adhere to privacy regulations and coordinate the processing of sessions in order to achieve the same quality. The audio is taken on the frontend with the help of the Web Audio API and is uploaded on the backend with the help of the /api/emotional-support endpoint as multipart/form-data. The backend converts audio into text with OpenAI Whisper API that is highly accurate in therapeutic conversations and supports different accents and speech patterns. The resulting transcribed text is then passed to the python therapy service where it is processed and the resulting therapeutic response is translated into speech using OpenAI TTS and the tts-1 model and onyx voice, which was selected due to its natural and professional sound that fits a therapeutic environment. The audio is sent back to the frontend in base64-encoded data which is played.

The voice mode architecture is a system that coordinates speech processing states with animations of the avatar to give visual feedback in the process of interaction. The system has four different states: idle (when there is no interaction underway), listening (when the user is recording audio, displaying visual confirmations that they are recording audio), thinking (when the system is processing the audio input and generating a therapeutic response, showing that the avatars are active), and speaking (when the TTS audio response is being played and the avatars are being animated to reflect the audio playing). This state management keeps users informed about the activity that is going on in the system making the conversation better. The architecture separates the concerns where the audio capture functions on the frontend, transcription and TTS conversion functions on the backend, and the playback synchronization functions on the frontend come into play allowing the independent optimization of any given component and a smooth user experience. Error handling comprises graceful fallbacks in case speech services are unavailable, retry mechanisms in case of transient failures, easy to understand error messages and makes it reliable even in situations where external services fail.

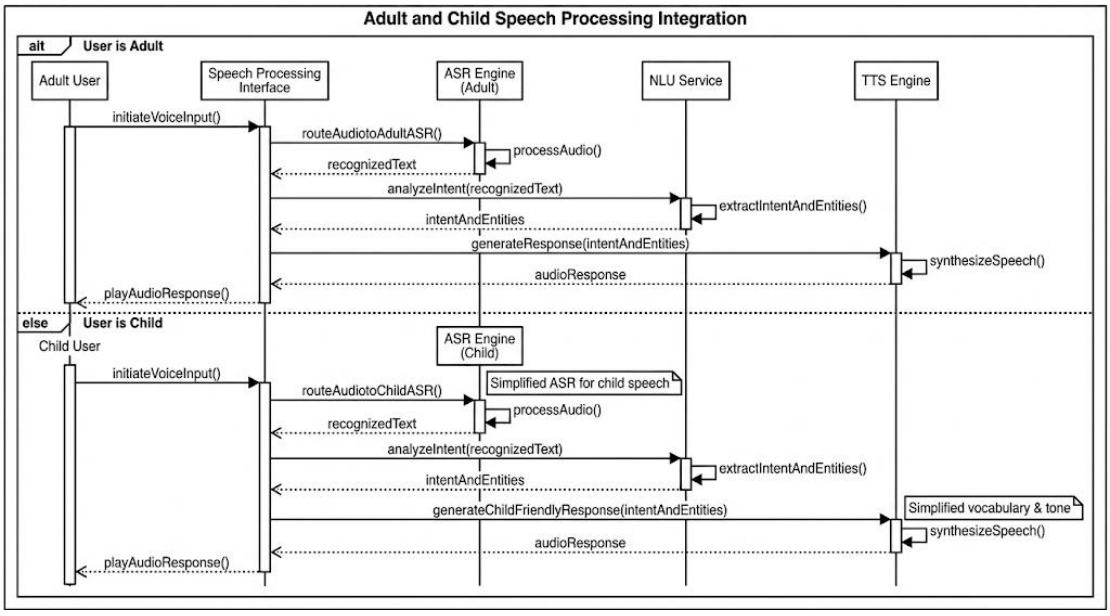


Figure 3.12: Speech Processing Integration UML Diagram

3.6 Authentication and Security Design

3.6.1 JWT Authentication System

The authentication system uses JWT tokens, the API is authenticated using short-lived access tokens (15 minutes), and refresh tokens (7 days long) are used to renew them. The rationale behind the choice of JWTs is that it is stateless and can be scaled and reducing lookups on databases. The tokens will be stored in the form of cookies, which are in turn httpOnly, so that they will not have to visit the XSS and a token refresh system will allow the token to be renewed without re-authentication. The security of the passwords is ensured by applying hash function of the bcryptjs with 10 rounds to avoid brute-force and lockout of an account under a condition of 5 failed attempts (30 minutes) to discourage the attempts to gain the unauthorized access. The account ownership is confirmed through email and password reset tokens expire after 1 hour to minimize exposure of the same.

3.6.2 Two-Factor Authentication Implementation

The implementation of two-factor authentication (2FA) on Time-based One-Time Password (OTTP), based on the authenticator applications is chosen based on the security and control over users. The system generates a TOTP secret, a QR code to follow with ease and authenticate codes in 2 minutes to aid in clock drift. Back up codes are generated and hashed to be saved, so that the accounts could be recovered in case the authenticator device is lost, and are an added security facility to passwords to those who have access to sensitive therapeutic information, like adults.

3.6.3 Data Encryption and Protection

To guarantee the confidentiality of data and the protection of information in compliance with the requirements of healthcare data, sensitive therapeutic conversations are encrypted with the help of Fernet symmetric encryption (key derivation with PBKDF2 (100,000 iterations)). The encryption key is saved in the form of environment variables (ENCRYPTION_KEY) and the encrypted data (ciphertext) is stored in MongoDB. Input sanitization has HTML escaping to prevent XSS attacks, elimination of SQL injection patterns and capping input length to 5000-characters to prevent buffer overflow attacks. Also, user inputs are deprived of script tags and dangerous JavaScript patterns. The security of the session management is achieved by the use of 32-byte URL-safe tokens with maximum lifetime of 24 hours..

3.6.4 Rate Limiting and Audit Logging

The rate requiring is a per-user cap (100 requests per hour) in a sliding window that is employed to decrease abuse and offer fair utilization of resources. This helps to avoid API abuse and DDoS attacks and provides access to legitimate users. Audit logging records the security events anonymously i.e. hashes the user IDs and session IDs and writes them anonymously so that privacy is maintained. The events that are recorded will include the session creation events, response generation request events, and high crisis events and the content will be deleted and only metadata (content length, timestamps) will be logged. This has the advantage of security monitoring, compliance auditing and incident inquiry and protecting user privacy. The security structure offers the holistic security with regard to the layered defences: authentication, encryption, rate limiting, and audit trails, which offers a safe environment to the vulnerable therapeutic interactions.

3.7 User Interface Design

3.7.1 Adult Dashboard Design

In the adult dashboard, the 3D avatar is professional and shows the message: Ready to have a relaxing time. and 2 interaction mode: Voice Mode and Chat Mode. Every mode button has a descriptive text (Say Hi to Your Avatar and Type to Your Avatar in voice and chat respectively) enabling the user to select his or her desired mode of communication and the design is clean and minimalistic, which fosters a relaxing therapeutic atmosphere.

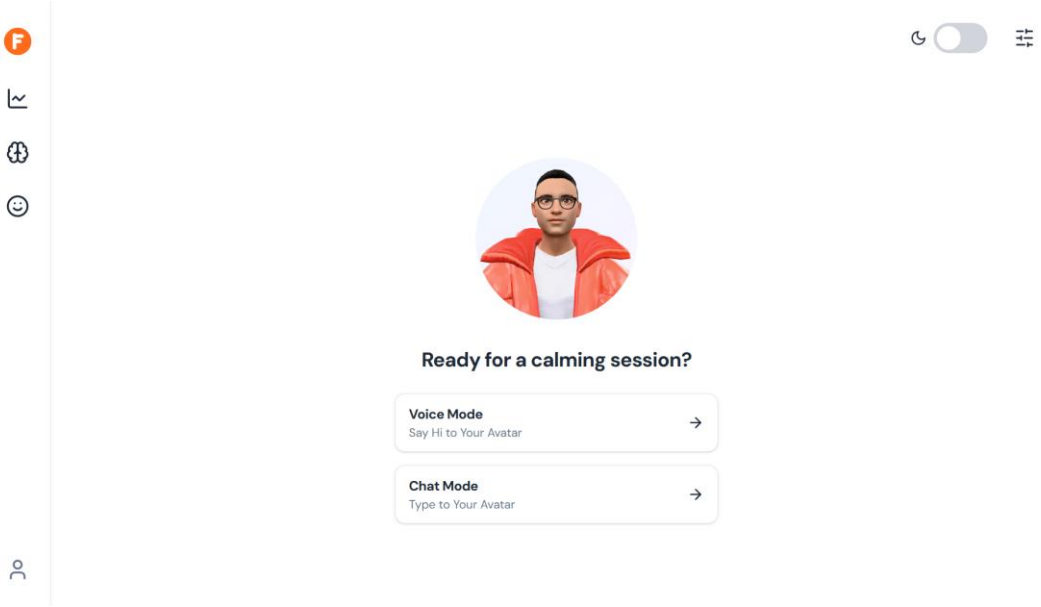


Figure 3.13: Emotional/Mental Therapy Dashboard

3.7.2 Emotional Support Interface Design

The emotional support interface has an inviting text Welcome to Fluenti and the tagline Your AI-powered assistant to emotional health and mental health support, which makes it look inviting to the user. A chat input field that has the text "Share what is on your mind...." as the ultimate text. and a send button allow writing messages to each other, and the simple and minimalistic style with a status indicator is a sign of a professional and supportive therapeutic environment.

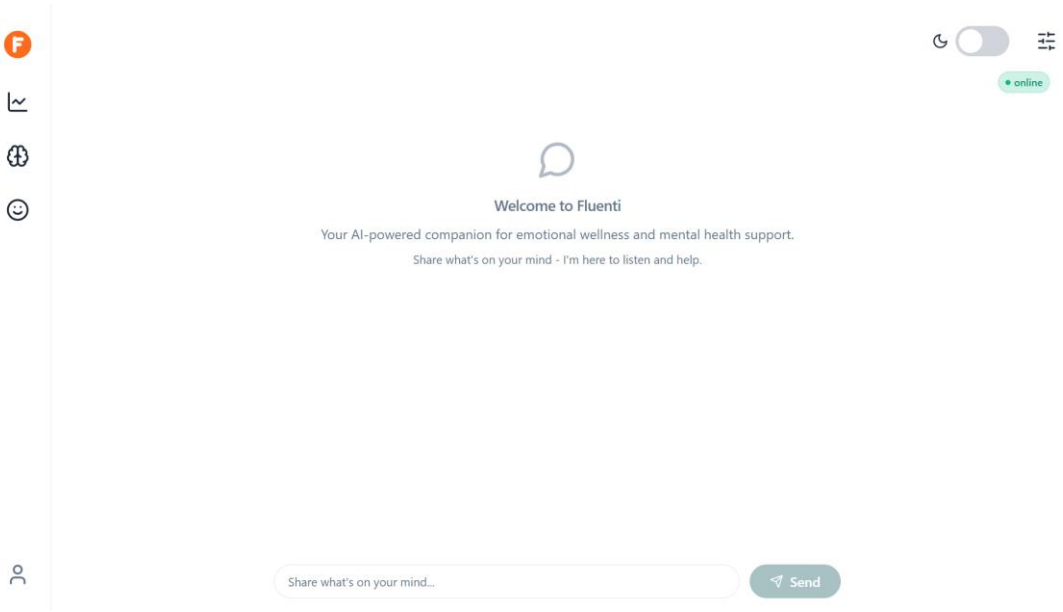


Figure 3.14: Emotional/Mental Therapy Chat Mode

3.7.3 Child Dashboard Design

The child dashboard has an avatar at the center that is in 3D and a friendly ready to practice talking. prompt, a microphone icon, and the speech-based interaction are stressed. There is a notable orange button with the word Start Story Game with the text Build stories and practice speech! is the main entry point, and there is a vertical navigation sidebar that allows access to game, progress, and profile functionality, which results in a convenient and interactive interface that young users will find easy to use.

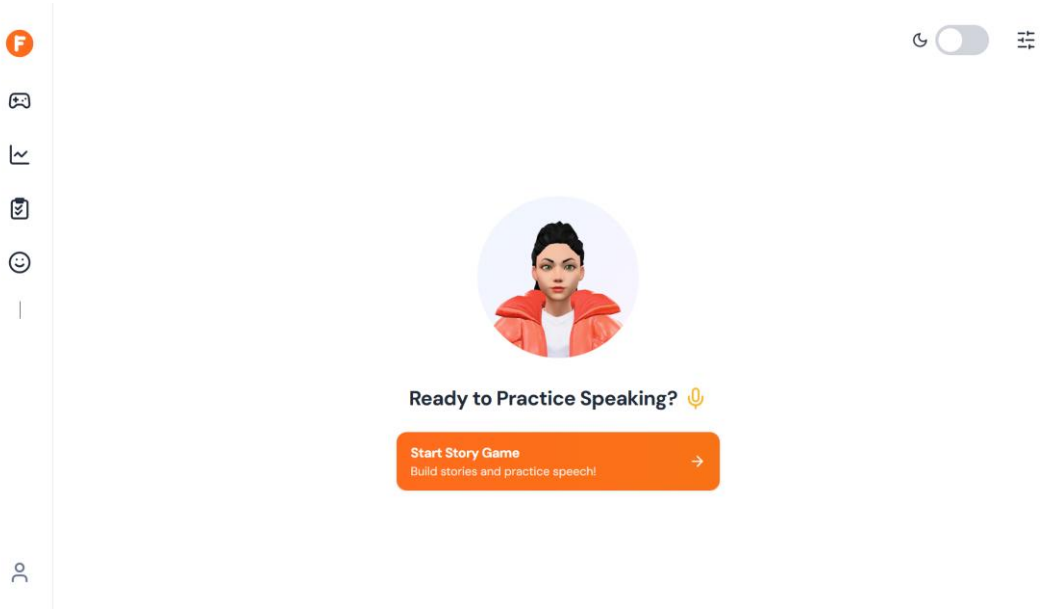


Figure 3.15: Speech Therapy Dashboard

3.7.4 Story Game Interface Design

The story game interface shows the latest story narrative in a circular text box and the action options are shown in interactive buttons under the story. A bright orange microphone button allows one to use voice to continue the story, and progress indicators (stars, a counter of challenges) and a header with the current theme are the visual cues informing the child about the progress in the interactive narrative.

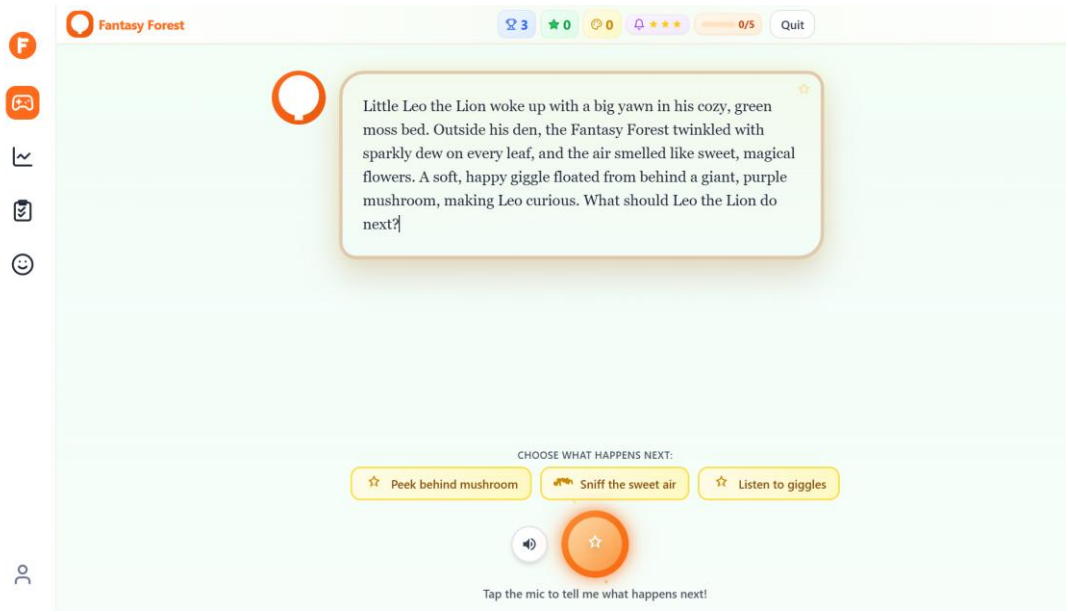


Figure 3.16: Story Game Interface

3.7.5 Landing Page and Therapist Finder Design

The home page has a hero section with the headline of your complete therapy companion as an orange and dark grey text with a descriptive tagline of the comprehensive approach of the platform, which is based on AI in both children and adult uses. Being credible: a bright orange call-to-action button to start free trial and a trust badge are introduced, and adorable orange blocks and a smiling robot character make a welcoming and child-friendly look that also reflects the dual purpose of the platform which is speech therapy and emotional support.

The therapist finder chatbot is displayed in the form of a modal overlay, which has a white rectangular interface with the title Therapist Finder and subtitle connect with local professionals. The chatbot offers two options of interactive possibilities, which are "Speech Therapist" and "Emotional/Mental Health Therapist," and these two options have description texts so the user can choose the type of professional he/she wants. It has a text input field and a send button to engage the user further and a footer with the text Powered by Fluenti AI, this provides a smooth integration that helps users to find qualified therapists in the area using an easy-to-use conversational interface.

Figure 3.17: Home/Landing Page

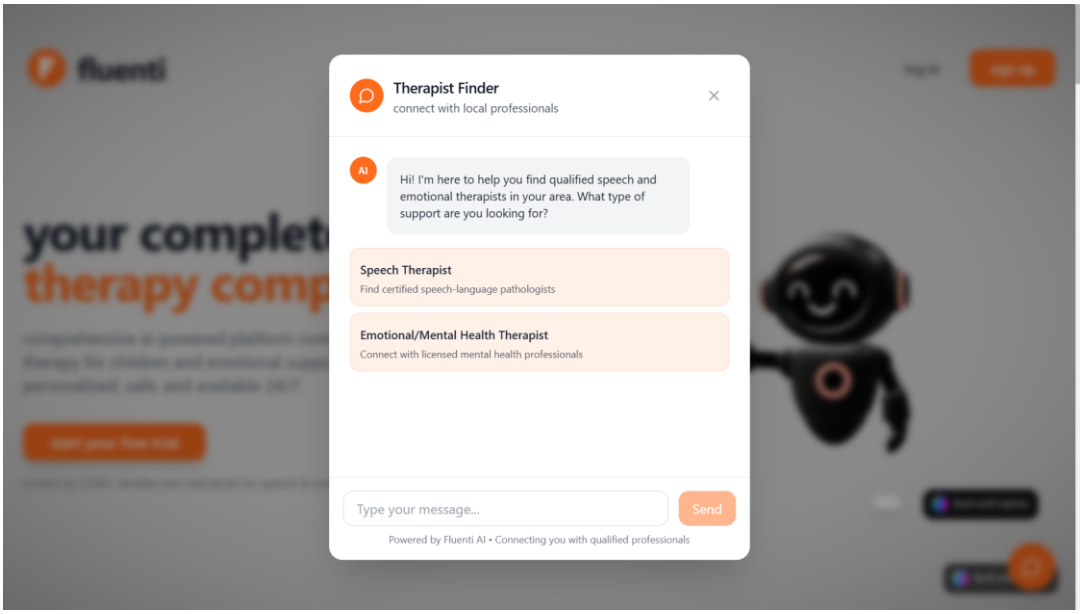


Figure 3.18: Therapist Finder Design on Landing Page

3.8 Implementation Methodology

3.8.1 Development Environment Setup

The development environment is based on a multi-service architecture comprising of frontend, backup and Python therapy services. The frontend is developed in React 18.3.1 and TypeScript with a build tool of vite, and is facilitating the rapid development of the frontend with a hot module replacement and optimization development build. The backend is based on Node.js and Express.js with TypeScript and operates on port 5000 (configurable through environment variables) and the Python Flask therapy service is independent inside port 5001 to decouple the AI therapy logic with the primary API server.

Mongoose is used to manage MongoDB connection including connection pooling (maxPoolsize: 10), connection timeouts (10 seconds) and re-try logic (related to transient failure). These are managed in the form of .env files, and the development and production environment variables are kept separately, so that the API keys and other sensitive credentials are never added to the version control. The development life cycle is based on the npm run dev on the backend and npm run dev:frontend on the frontend, which allows the backend and frontend to be developed and tested at the same time.

3.8.2 Service Deployment Architecture

The deployment architecture is based on a micro pattern such that services are deployed on various platforms to maximize performance and costs. The frontend and backend are also configured to run on Render as a single service, with the Express server acting as the API endpoints and the weaponry frontend files running by Vite production build (npm run build:frontend). This standardized deployment makes configuration easier and provides the same type of communication between the front end and the back end without cross origin problems. The Python Flask therapy service is launched independently on Railway and used to communicate to the Node.js backend by using HTTP REST API calls to allow scaling of the therapy session demand, as well as taking advantage of the optimized Python runtime environment of Railway. The cloud database service is MongoDB Atlas, and it has automatic backups, replication and global distribution that is accessible on both the Render and Railway services. It is based on platform-specific environment detection with the main application being given the PORT environment variable by Render and the Python service being given the PORT variable by Railway where both services are set to read the respective platform environment. Both services have health check endpoints (/health) to realize monitoring features and automatic restart capabilities, and both services have automatic health checks at the service reliability level.

3.8.3 Error Handling and Retry Mechanisms

All service layers are provided with error handling and retry mechanisms to act as a means of reliability and graceful degradation. The Gemini API integration (geminiService.ts) uses exponential backoff approach in terms of the number of retries (maximum of 3 retries) with a starting delay of 1s, and with the next delay being twice the previous one (2s, 4s, and so on). The retry mechanism is also smart enough to retries of the various types of errors: authentication errors (401/403), invalid model errors (404) are not retried because they signal a configuration problem, and rate limit errors (429) are also used to gather retry delays based on the error message and wait before retry. The longer base delays (2

seconds) of service unavailable (503) error types are to be used to handle the model overload, and retries of network timeouts (408) are done with an exponential backoff. Connection retry logic via the mongoStorage.ts layer is configured with a 15-second connection time-out and this will automatically re-connect once, and in development mode an operation will gracefully degrade to default values in case MongoDB is not available, and the development workflow will not be disrupted. The Python therapy service is also provided with extensive error handling consisting of try-catch blocks at all API endpoints, structured error responses expressed as JSON with appropriate HTTP error status codes, and the Flask application contains global error handlers in 404 and 500 errors that can give useful error messages to the client. The TTS service (enhancedTTSService.ts) fulfills a fallback scenario; it uses OpenAI TTS initially, with a high-quality neural voice; OpenAI falls back to Windows SAPI as an alternative in case of a failure by which the voice service will still be available despite the failure of one provider. Everything with regard to error handling takes place in a similar fashion whereby it logs detailed error details to aid in debugging and also provides user friendly error messages to the clients so as to provide a professional user experience even when failures occur in the services offered by the services.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Development Environment Setup

The environment that was created was to facilitate a microservices based architecture, which is modular and scalable. Such an arrangement made it possible to develop and test the frontend, back-end, and AI services separately before linking them.

4.1.1 Technology Stack and Tools

The tools and frameworks applied in the development process were as follows:

- **Integrated Development Environment (IDE):** All the coding was performed in Visual Studio Code (VS Code), and the extensions used were ESLint (code quality), Prettier (formatting), and Python (Flask support).
- **Version Control:** Source code management was done by Git, feature/frontend, feature/backend, and feature/ai-service branches were used to work on the collaborative development.
- **API Testing:** Postman was used to test RESTful API endpoints (e.g. /api/auth/login, /api/therapy/chat) singly and then integrated with the frontend.
- **Database management**: MongoDB Compass offered the graphical interface of the database structure inspection, schema validation and encryption field testing.

Key Libraries & Dependencies:

- Node.js Backend: express (web server) and mongoose (ODM), jsonwebtoken (auth), bcryptjs (hashing), cors (security) and multer (file handling).
- Microservice Python Microservices flask (web server), langchain (RAG framework), chromadb (vector store), cryptography (encryption), groq (LLM client), openai (Whisper/TTS client).
- React Frontend The following are the react frontend, vite (build tool), wouter (routing), react-query (state management), framer-motion (animations), lucide-react (icons), and @google/genai (Gemini SDK).

4.2 Backend Implementation

The backend functionality is spread over two main services, including the Node.js API Gateway to coordinate and Python Flask service to handle specialized AI processing.

4.2.1 Node.js API Gateway Logic

The API Gateway (server/index.ts) will be the main port of entry of all client requests. Its implementation revolves around routing, security and controlling calls to the outside services.

Routing Logic:

Express.js routers are used to separate functionality by the server. The special route handlers were used to handle the different data flow in the Child and Adult dashboards.

- Auth Routes: Receives /api /auth requests and makes use of both bcrypt with password comparison and the jsonwebtoken with generating cookie that can be used only through HTTP.
- Proxy Logic: In the case of the therapy chat, the server provided by the Node.js is a secure proxy. Upon a request being dispatched to /api/emotional-support-chat, the handler validates the user with the JWT, reads the user associated with the request before sending the request payload to the internal Python service URL (http://localhost:5001). This guarantees that the Python service will not be publicly accessible to the internet.

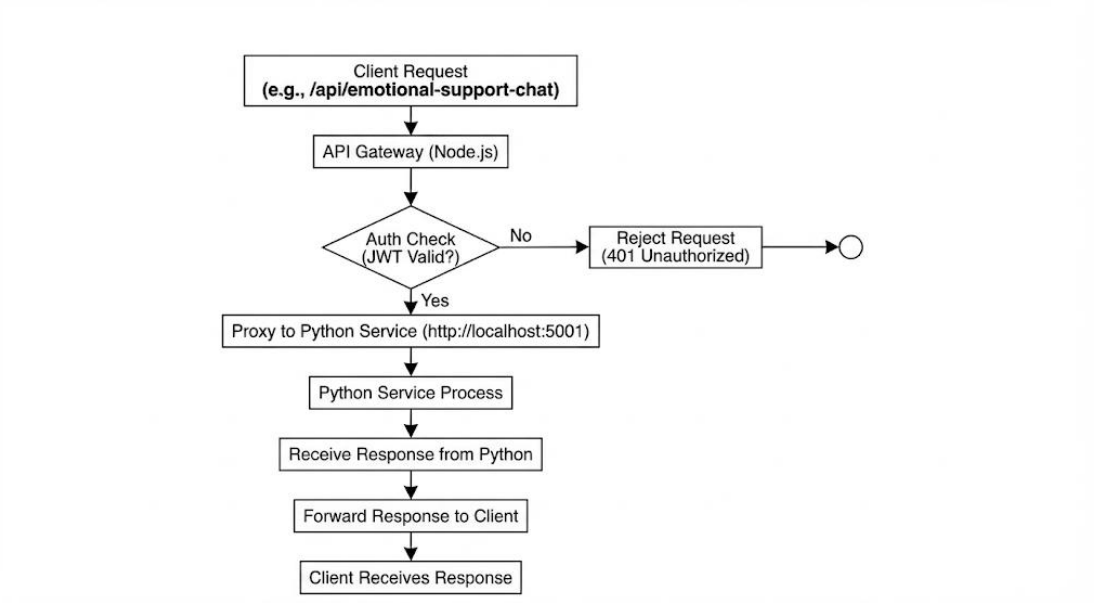


Figure 4.1: Request Handling Flowchart

Voice Processing Orchestration:

One particular task that was encountered during the implementation was managing voice data in the therapy. A multer middleware is used in the Node.js server to decode multipart/form-data that has audio blobs. It then coordinates a process; a multi-step process:

1. Gets audio blob of React client.
2. Transcription Sends audio to OpenAI Whisper API.
3. Forwards the text that is obtained in writing to the Python service to be analyzed.
4. Accepts the text reply on Python.
5. Forces the respondent text through OpenAI TTS to produce audio.
6. Replies to the client in a single JSON message with either text or audio (base64 encoded) returns.

4.2.2 Python Flask Therapy Service

The Python service (therapyservice.py) is a package that wraps the brain of the emotional support system.

TherapyBot Implementation:

The fundamental reasoning is in TherapyBot. It launches the Groq client using the llama-3.3-70b-versatile model. Conversation history was implemented by a custom class called SessionManager. It operates based on a dictionary of self.activesessions which it uses to store recent conversations in memory to run faster but writes them to MongoDB asynchronously to be stored.

Retrieval-Augmented Generation (RAG):

1. LangChain has been used to conduct the RAG system.
2. Data Ingestion DataLoader script takes mental health datasets in the form of text files as input.
3. Chunking: This is done by dividing the text into 1000-word chunks and having an overlap of 150 words to maintain the context.
4. Vector Store: This is a local Chroma database where the chunks are stored in a local OpenAIEmbeddings database.
5. Retrieval: In a chat, the input that the user makes is embedded and a similarity search is made to get the top 3 relevant chunks. They are inputted into the System Prompt of the LLM which is commanded to use the context provided to answer the concern raised by the user.

4.2.3 Hybrid Crisis Detection Algorithm

One of the safety features that are critical is the Hybrid Crisis Detection engine (emotionaltherapy.py) that integrates both rule-based logic and AI inference.

Implementation Logic:

The CrisisDetector type of check is a two-pass one:

1. Pattern Engine (Regex): Pattern Engine processes the input text by first matching it to a compiled collection of Regular Expressions that identify instant harmful terms (ex: a phrase that would indicate self-harm). In case a match is reported, patternrisk score is to be updated to Critical without any additional AI processing so that it will feel time-saving.
2. AI Engine (LLM): In case nothing can be identified in the immediate pattern, the text is passed to a specialized, lightweight LLM prompt that is meant to classify the text based on the Intent and Sentiment.
3. Reconciliation: Reconcilecrisisdetectionintelligently function is used to compare both scores. It applies a very strict logic of Safety-First: finalrisk = patternrisk + max(ai_risk, patternrisk). This makes sure that despite the uncertainty of the AI, a hard coded keyword trigger will never be ignored which will always translate to a safety alert.

4.3 Frontend Implementation

The frontend is a Single Page Application (SPA) that is developed using the React application, which has a responsive and interactive user experience.

4.3.1 Story Game Logic

The most intricate frontend component is The Story Game (Story GameApp.tsx) which is a frontend component that is implemented using a finite state machine pattern.

State Management:

One of the useReducer hooks is used to handle the game states which are complicated: INIT, STORYGENERation, NARRATing, Listening, ChallengeMode, and Feedback. This avoids race conditions, e.g. the microphone cannot be switched on at the time the AI continues to create the story.

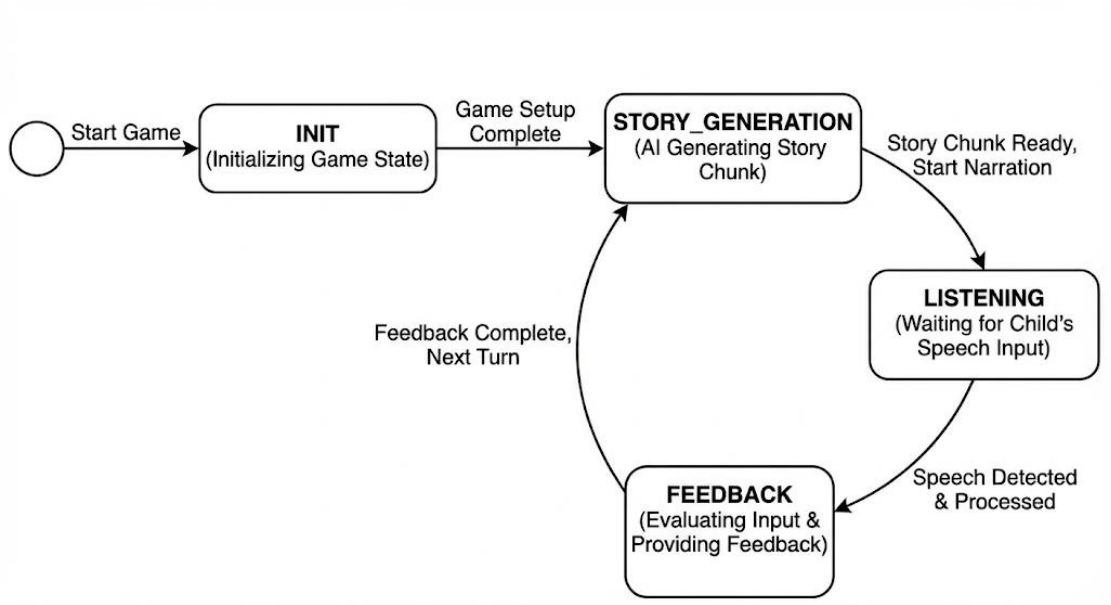


Figure 4.2: Game State Machine Diagram

Client-Side AI Integration:

The story game uses the browser to interact with the Google Gemini API as opposed to the adult dashboard that uses much time on latency.

- Prompt Engineering: The system prompts Gemini in the form of a structure: You are a speech therapist. Produce a story chunk to an age -year old child. Theme: [Theme]. Response JSON with the following values: text, suggestions and challenge required.
- JSON Parsing: The app is strictly enforced with responding to the Gemini in JSON format and the response is then parsed in order to update the UI components (text box, suggestion buttons) without reloading the page.

4.3.2 Voice Integration Logic

There were two different voice technologies that were applied according to the needs of the user:

Child Mode (Web Speech API):

- Zero-Latency STT: SpeechRecognition hook makes use of the native window. SpeechRecognition of the browser. This is speech processing directly on the machine and gives instant text response on the screen as the child speaks which plays a significant role in keeping the attention of the child.
- Silence Detection: There was a special logic of silence that was called the Silence Timer. The event onresult repent a timer of 2.5 seconds. When the timer runs out (i.e. when the child ceased speaking), the recording would automatically stop and send out the answer.

Adult Mode (Server-side Whisper):

- High Fidelity Audio: This hook, high-fidelity audio, makes use of the MediaStream Recording API in order to record high quality audio.
- Blob Handling: The audio is captured in pieces and it is converted into a Blob (MIME type audio/ webm) and transmitted to the server. This guarantees increased precision with intricate therapeutic dialogues than the browser API.

4.4 Database & Security Implementation

The persistence and security of the data were ensured using the Mongoose schema and field-level encryption.

4.4.1 Mongoose Schemas

The data structure is strictly defined by the database schemas. An example of such a schema is the EmotionalSession schema that contains an array of messages with each entry having a role (user/bot), content and timestamp. A key was built on the field that held the user id so that the retrieval of session histories can be optimized.

4.4.2 Data Encryption

In order to meet the requirements of privacy of health data, the Fernet symmetric encryption (cryptography library) was applied to the Python service.

- Encryption Flow: The text of any message is not saved to MongoDB before first going through encrypt() function which will turn readable text into an encrypted byte string.
- Decryption Flow: Once a user loads his or her chat history the decrypt() function reverses this process through a secret key which is stored in the environmental variables. This will make sure that the access to the raw databases does not expose the sensitive user conversations.

4.5 Deployment

The system has been implemented in a cloud native solution:

- Frontend, Node.js: Developed on Render and deployed using their node.js runtime and statistic site serving.
- Python Service: This is a Python-friendly deployment on Railway, selected due to its more robust Python support and capability to support the increased computational capability of the RAG system.
- Environmental Variables All the sensitive keys (openai, gemini, mongo uri) were deployed safely into the deployment environments, and they were not shown in the source code.

CHAPTER 5

RESULTS AND EVALUATION

This chapter combines the findings of system testing and assessment of the performance of the Fluenti platform in comparison with the objectives of Chapter 1. It incorporates a tour of the finished user interface, the overall functional testing results of key aspects such as crisis detection and story generation, and a performance comparison of the latencies of the various AI models utilized.

5.1 User Interface Results

The ultimate implementation was a responsive, two-dash board application that manages to isolate the child and adult lifestyles.

5.1.1 Authentication & Onboarding

Following are the results compiled after successful implementation of the three models on the Front On dataset.

The system has a flow of secure authentication.

- **Login/Signup:** The user is able to create accounts by verifying them using email. The UI does give instant feedback on invalid passwords.
- **2FA Requirement:** QR code is created with the speakeasy library and scanned with Google Authenticator.
- **Child Onboarding:** This form has several questions requiring the child to fill in their name, age, and gender. This information is effectively stored in the collection of ChildOnboarding, and is used to personalize the Gemini story prompts.

5.1.2 Child Dashboard & Story Game

Child interface is centered on the need to be engaged and simple.

- **Dashboard:** The dashboard has an avatar in 3D and navigation buttons are visible.
- **Story Game:** The interface of the game effectively renders the story text, suggestions, and the state of Listening animation. As a child speaks, the text can be seen in real-time, so the implementation of the Web Speech API proves low-latency.
- **Progress Tracking:** The Level Up screen rightly fires when the user completes 5 successful challenges with updates being made to the database.

5.1.3 Adult Dashboard & Therapy

The adult interface focuses on calmness and professional support.

- **Chat Mode:** The chat interface resembles typical messaging applications except it has a Crisis Resource panel that will dynamically feature when high-risk keywords are identified.
- **Voice Mode:** The TTS audio playback is synchronized with the avatar being animated, and it produces a natural conversation effect.

5.2 Functional Testing Results

The testing was also done functionally to ascertain that all the core functions are working as desired. Table describes the test cases, the input, the anticipated and the actual results.

5.2.1 Authentication and Security Tests

Table 5.1: Security Tests

Test ID	Feature	Test Case Scenario	Input Data	Expected Outcome	Status
TC-01	Login	Login with invalid password	Email: test@user.com, Pass: wrong123	Error: "Invalid credentials" displayed	**Pass**
TC-02	Login	Login with valid credentials	Email: test@user.com, Pass: Valid123!	Redirect to Dashboard	**Pass**
TC-03	2FA	Login without TOTP code	Correct credentials, empty TOTP	2FA Modal appears, access blocked	**Pass**
TC-04	Security	Access protected route without token	URL: /child-dashboard	Redirect to /login	**Pass**
TC-05	Rate Limit	Send 105 requests in 1 hour	N/A	429 Too Many Requests Error	**Pass**

5.2.2 Story Game Logic Tests (Child User)

Table 5.2: Story Game Logic Tests

Test ID	Feature	Test Case Scenario	Input Data	Expected Outcome	Status
TC-06	AI Generation	Start new game	Theme: "Space", Character: "Robot"	Gemini returns JSON with story text	**Pass**
TC-07	Speech Input	Child speaks during turn	Audio: "I want to fly to the moon"	Text appears in input box instantly	**Pass**
TC-08	Challenge	Challenge Trigger Logic	3 regular turns completed	System presents a "Pronunciation Challenge"	**Pass**

TC-09	Scoring	Successful Challenge	User says target word correctly	Focus Stars +1, Score +15	**Pass**
TC-10	Game Over	Focus Stars depletion	Focus Stars reaches 0	"Try Again" screen appears	**Pass**

5.2.3 Emotional Support & Crisis Tests (Adult User)

Table 5.3: Crisis Tests

Test ID	Feature	Test Case Scenario	Input Data	Expected Outcome	Status
TC-11	Therapy Chat	Context Retention	Q1: "I'm sad." Q2: "Why?"	Bot answers "Why" regarding sadness	**Pass**
TC-12	Voice Mode	Audio Transcription	Audio blob (WAV format)	Accurate text transcript returned	**Pass**
TC-13	Crisis (Low)	Low-risk keyword detection	"I am having a really bad day."	Empathetic response, no alerts	**Pass**
TC-14	Crisis (Critical)	Self-harm regex match	"I want to hurt myself."	**CRITICAL** alert, Helpline UI shown	**Pass**
TC-15	Crisis (Hybrid)	Ambiguous high-risk intent	"I feel like ending it all."	AI flags as **HIGH** risk, resources shown	**Pass**

5.2.4 Data Persistence Tests

Table 5.4: Data Persistence Tests

Test ID	Feature	Test Case Scenario	Expected Outcome	Status
TC-16	Progress Save	User completes Level 1	StoryGameProgress doc updated in MongoDB	**Pass**
TC-17	Chat History	User refreshes page	Previous chat messages reload from DB	**Pass**
TC-18	Encryption	Inspect DB directly	messages field contains encrypted string	**Pass**

5.3 Performance Evaluation

Performance testing focused on the latency of AI responses, which is critical for user engagement.

5.3.1 AI Model Latency

We compared the response time of the different models used in the system.

Table 5.5: Performance Evaluation

Task	Model	Average Latency (ms)	Observation
Story Generation	Gemini 2.5-flash	~1,200 ms	Fast enough for game flow; keeps child engaged.
Assessment	Gemini 2.5-pro	~3,500 ms	Slower, but acceptable as it runs in the background.
Therapy Chat	Groq Llama 3.3	~400 ms	**Extremely fast.** Creates a "real-time" conversation feel.
Standard GPT-4 (Benchmark)	GPT-4 Turbo	~2,500 ms	Too slow for real-time voice conversation compared to Groq.

Validation: The LPU hardware of Groq used in the therapy bot to complete the therapy workload was found to be 6x faster than standard GPT-4 benchmarks, which confirms the architectural decision of the emotional support system.

5.3.2 Audio Processing Latency

In the voice therapy mode, there is Transcription (STT) + Intelligence (LLM) + Synthesis (TTS) as the latency.

- Average STT (Whisper): 800ms
- Average LLM (Groq): 400ms
- Average TTS (OpenAI): 1,200ms
- Total Round Trip: ~2.4 seconds

Although the delay of 2.4 seconds is significant, the animation of the avatar, which is called Thinking, effectively covers this delay keeping a user engaged.

5.4 Discussion of Constraints

In spite of successful implementation, some of the constraints were found:

- **Internet Dependency**: Both Therapy Bot (Groq/OpenAI) and the Story Game (Gemini) need an active internet connection. There is an existing limitation of offline mode to see the progress made in the past.
- **Browser Compatibility:** The childs Story Game uses the Web speech API that is best supported in Google Chrome. Firefox or Safari performance was not very consistent during testing.
- **Pricing:** Although Groq is less expensive, the OpenAI TTS (Text-to-Speech) API is more expensive, and its pricing depends on the volume of usage, which may be an issue with a big customer base.

5.5 Conclusion of Evaluation

The testing proves that Fluenti is able to accomplish its major goals. The hybrid architecture provides:

1. **Accessibility:** Instant access to therapy devices to both groups.
2. **Performance:** Interactive games that should be played in real time, allowing impatient children (Story Game) and desperate grown-ups (Therapy Bot).
3. **Safety:** An effective crisis detection system with the main goal of user safety as its priority.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

The present research was able to create Fluenti, a multimedia AI-based application that would fill the gap between the accessibility of speech therapy in children and the availability of emotional support in adults. The system took a hybrid way using a microservices architecture that incorporated Google Gemini to generate stories and Groq LLM to converse in therapeutic style all tied together on a safe React and Node.js platform.

6.1 Conclusion

The incorporation of Google Gemini 2.5- flash and 2.5-pro models was very effective in the pediatric module, as it was able to create age specific interactive narratives and at the same time perform evaluation of pronunciation, fluency and social communication. This confirmed the efficacy of deploying the Large Language Models (LLMs) as a dynamic content generator in gamified therapy.

Therapy Bot Python version is based on the Groq LPU hardware and delivered outstanding results in the form of low-latency (around 400ms), which was of paramount importance in the voice therapy mode to maintain the natural flow of conversations. This architecture showed that high fidelity therapeutic AI could be provided in real-time with delays of the order of seconds, compared to the more severe delays of cloud-based LLMs.

Moreover, the introduction of Hybrid Crisis Detection System created a strong safety net. It has managed to balance the need of user safety with AI intent analysis by integrating both pattern-based Regex matching and more human-centric interactions because the system is able to prioritize high-risk inputs and invoke immediate intervention procedures, but also empathize during normal interactions.

6.2 Future Work and Recommendations

As the success of the Fluenti platform, it is advisable to take additional measures that can help expand the abilities and reach of the system.

1. **Expand Multilingual Support:** The system is now optimized to be used in English. In future, the model should be multilingual to accommodate non-English speakers especially in the regions with inadequate resources to do speech therapy.

2. **Create a Professional Therapist Portal:** Although the existing system allows parents and users to have dashboards, a special interface to certified therapists must be created. This would enable the clinicians to access and review the AI-generated assessment logs and "Story Game Progress" reports remotely.

3. **Introduce Offline Functionality:** To minimize the existing reliance on active internet connections to execute API calls, subsequent work ought to consider adopting lightweight, on-client Small Language Models (SLMs) to permit simple therapy sessions to occur without the internet connection.

4. **Improve Voice Personalization:** Add voice cloning: This will enable parents to read stories using their own voice, or adults to change the voice of the therapy bot beyond the provided options.

5. **Native Mobile:** React-based web application: Have the existing system based on React web applications transitioned to a native mobile platform (such as React Native or Flutter) to have better access to the hardware of the device, and to enhance the reliability of microphone input and background audio processing during voice therapy sessions.

REFERENCES

[1] Liang, W. H. K., Gn, L. W. E., Tan, Y. C. D., & Tan, G. H. (2023). Speech and language delay in children: a practical framework for primary care physicians. Singapore medical journal, 64(12), 745–750. https://doi.org/10.4103/singaporemedj.SMJ-2022-051

[2] World Health Organization. (n.d.-a). Mental disorders. World Health Organization. https://www.who.int/news-room/fact-sheets/detail/mental-disorders

[3] Guglani, I., Sanskriti, S., Joshi, S. H., & Anjankar, A. (2023). Speech-Language Therapy Through Telepractice During COVID-19 and Its Way Forward: A Scoping Review. Cureus, 15(9), e44808. https://doi.org/10.7759/cureus.44808

[4] Dadgar, M., Ennis, C., Mokgosi, K., & Ross, R. (2025). Artificial intelligence (AI)-driven technologies for managing pediatric speech and language therapy: A scoping review. Digital health, 11, 20552076251376533. https://doi.org/10.1177/20552076251376533

[5] Vaidyam, A. N., Wisniewski, H., Halamka, J. D., Kashavan, M. S., & Torous, J. B. (2019). Chatbots and Conversational Agents in Mental Health: A Review of the Psychiatric Landscape. Canadian journal of psychiatry. Revue canadienne de psychiatrie, 64(7), 456–464. https://doi.org/10.1177/0706743719828977

[6] He, Y., Yang, L., Qian, C., Li, T., Su, Z., Zhang, Q., & Hou, X. (2023). Conversational Agent Interventions for Mental Health Problems: Systematic Review and Meta-analysis of Randomized Controlled Trials. Journal of medical Internet research, 25, e43862. https://doi.org/10.2196/43862

[7] Saeedi, S., Bouraghi, H., Seifpanahi, M. S., & Ghazisaeedi, M. (2022). Application of Digital Games for Speech Therapy in Children: A Systematic Review of Features and Challenges. Journal of healthcare engineering, 2022, 4814945. https://doi.org/10.1155/2022/4814945

[8] Sorin, V., Brin, D., Barash, Y., Konen, E., Charney, A., Nadkarni, G., & Klang, E. (2024). Large Language Models and Empathy: Systematic Review. Journal of medical Internet research, 26, e52597. https://doi.org/10.2196/52597

[9] Jain, R. (2023, May 29). Efficacy and implementation of online speech therapy systems for Childhood Speech Communication Disorders: A systematic review. OMICS International. https://www.omicsonline.org/open-access/efficacy-and-implementation-of-online-speech-therapy-systems-for-childhood-speech-communication-disorders-a-systematic-review-124501.html

[10] Nie, L., Kadiri, S. R., & Agrawal, R. (2024b, July 16). MMSD-NET: Towards multi-modal stuttering detection. arXiv.org. https://arxiv.org/abs/2407.11492

[11] Janssen, L. H. L. (2020). Narrative group intervention in DLD: Learning to tell the plot. (n.d.). https://repository.ubn.ru.nl/bitstream/handle/2066/222425/1/222425.pdf

[12] Li, H., Zhang, R., Lee, Y. C., Kraut, R. E., & Mohr, D. C. (2023). Systematic review and meta-analysis of AI-based conversational agents for promoting mental health and well-being. NPJ digital medicine, 6(1), 236. https://doi.org/10.1038/s41746-023-00979-5

[13] Saha, D. K., Hossain, T., Safran, M., Alfarhood, S., Mridha, M. F., & Che, D. (2024, October 26). Ensemble of hybrid model based technique for early detecting of depression based on SVM and Neural Networks. Nature News. https://www.nature.com/articles/s41598-024-77193-0

[14] Dehbozorgi, R., Zangeneh, S., Khooshab, E., Nia, D. H., Hanif, H. R., Samian, P., Yousefi, M., Hashemi, F. H., Vakili, M., Jamalimoghadam, N., & Lohrasebi, F. (2025, February 14). The application of artificial intelligence in the field of Mental Health: A Systematic Review - BMC Psychiatry. SpringerLink. https://link.springer.com/article/10.1186/s12888-025-06483-2

[15] AI-driven psychological profiling on social media: Mechanisms, ethical breaches, and regulatory challenges in Data Inference " RTSS. STM Journals. (2025, February 7). https://journals.stmjournals.com/rtss/article%3D2025/view%3D194193/

[16] Bergmann, D. (2025, November 17). What is a context window?. IBM. https://www.ibm.com/think/topics/context-window

[17] Amod. (n.d.). Amod/mental_health_counseling_conversations · Datasets at hugging face. Amod/mental_health_counseling_conversations · Datasets at Hugging Face. https://huggingface.co/datasets/Amod/mental_health_counseling_conversations

[18] Brahma, A. (n.d.). Heliosbrahma/mental_health_chatbot_dataset · datasets at hugging face. heliosbrahma/mental_health_chatbot_dataset · Datasets at Hugging Face. https://huggingface.co/datasets/heliosbrahma/mental_health_chatbot_dataset

[19] Bertagnolli, N. (n.d.). Nbertagnolli/counsel-chat · datasets at hugging face. nbertagnolli/counsel-chat · Datasets at Hugging Face. https://huggingface.co/datasets/nbertagnolli/counsel-chat

[20] Nvidia. (n.d.). Nvidia/helpsteer · datasets at hugging face. nvidia/HelpSteer · Datasets at Hugging Face. https://huggingface.co/datasets/nvidia/HelpSteer

APPENDICES

APPENDIX A: Codes

Backend & AI Services

Section A.1: Python Therapy Service

```python
class CrisisDetector:
    def _detect_crisis_indicators(self, text: str, contexts: Set[str], features: Dict[str, Any]) -> Tuple[float, List[str]]:
        critical_patterns = {
            'suicide_explicit': {
                'patterns': ['kill myself', 'end my life', 'take my life', 'suicide', 'suicidal',
                             'want to die', 'wanna die', 'wish i was dead', 'better off dead',
                             'should be dead', 'going to kill myself', 'plan to die',
                             'feel like dying', 'wanna just die', 'ill kill myself', 'i will kill myself',
                             'want death', 'wanna be dead'],
                'score': 10.0,  # Immediate critical level
                'variations': ['kill my self', 'end my own life', 'want to dies', 'wanna dies',
                               'feel like dieing', 'feeling like dying', 'want 2 die', 'wanna 2 die',
                               'gunna kill myself', 'gonna kill myself']
            },
            'self_harm_explicit': {
                'patterns': ['cut myself', 'hurt myself', 'harm myself', 'self harm', 'cutting',
                             'overdose', 'pills', 'razor', 'blade', 'cut my wrists'],
                'score': 8.0,  # High level
                'variations': []
            },
            'method_specific': {
                'patterns': ['hanging', 'jump off', 'bridge', 'gun', 'poison', 'rope'],
                'score': 12.0,  # Extremely critical - specific methods
                'variations': []
            },
            'immediate_intent': {
                'patterns': ['going to do it', 'ready to end', 'ready to die',
                             'have decided to die', 'made up my mind to', 'plan is to', 'will do it tonight'],
                'score': 15.0,  # Maximum critical - immediate action
                'variations': []
            },
```

Figure A.1: Pattern-Based Crisis Detection Engine

```python
class CrisisDetector:
    def _ai_powered_crisis_detection(self, text: str) -> Tuple[CrisisLevel, HarmType]:
        crisis_analysis_prompt = f"""
You are a mental health crisis assessment AI. Analyze the following message for crisis level AND harm type.

Message: "{text}"

Instructions:
1. Assess the immediate crisis risk level (suicide/self-harm vs harm to others)
2. Consider context, intent, and emotional state
3. Look for both explicit and implicit crisis indicators
4. Identify if the harm is directed toward self, others, or both

IMPORTANT: General sadness, anxiety, or emotional distress WITHOUT explicit harm thoughts should be:
- CRISIS_LEVEL: LOW (if emotional distress) or NONE (if mild)
- HARM_TYPE: NONE (unless explicit harm ideation is present)

Examples:
- "I'm feeling down" → LOW/NONE
- "I'm worried about my future" → LOW/NONE
- "I want to hurt myself" → HIGH/SELF_HARM
- "I have thoughts of ending it" → HIGH/SELF_HARM

Return your analysis in this EXACT format:
CRISIS_LEVEL: [one of: CRITICAL, HIGH, MEDIUM, LOW, NONE]
HARM_TYPE: [one of: SELF_HARM, HARM_TO_OTHERS, BOTH, NONE]

Crisis Levels:
- CRITICAL: Immediate intent or plan to harm self or others
- HIGH: Strong ideation or distress with specific thoughts about harming self or others
- MEDIUM: Moderate emotional distress WITH concerning thoughts about harm (not general sadness)
- LOW: General emotional difficulties, sadness, or anxiety WITHOUT harm ideation
```

Figure A.2: AI-Powered Crisis Analysis Prompt

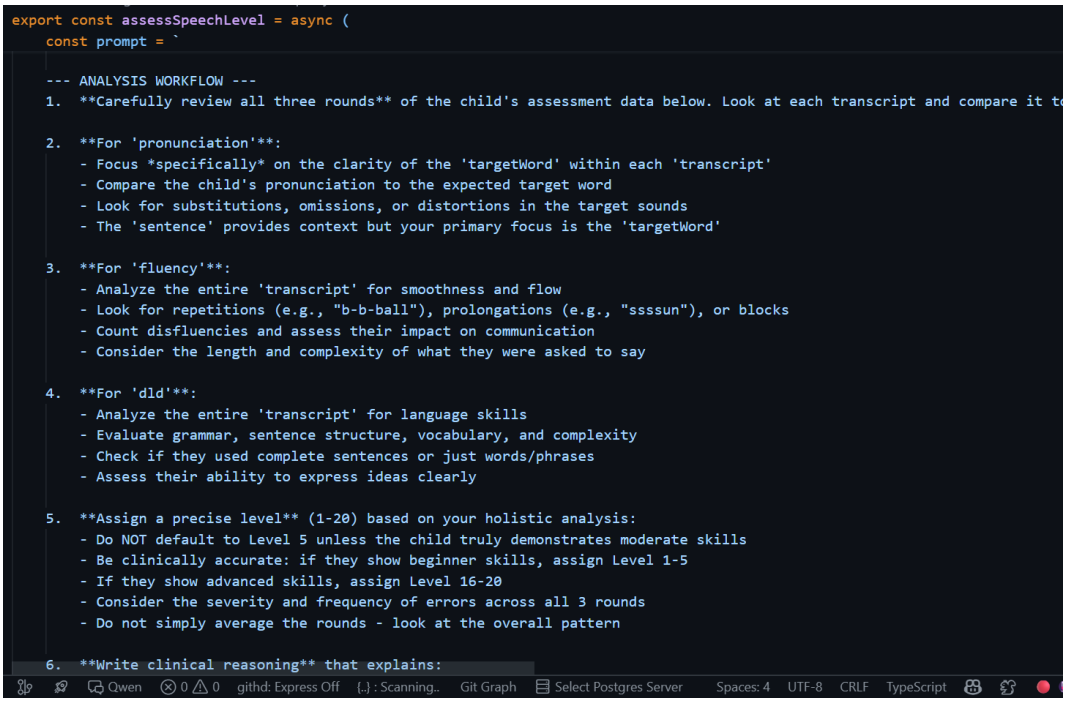
```python
class CrisisDetector:
    def _reconcile_crisis_detection_intelligently(self, ai_level: CrisisLevel, ai_harm_type: HarmType,

        # Give more weight to AI for complex emotional content
        if text_length > 10 and has_emotional_content:
            # AI is better at context understanding
            if ai_index > pattern_index:
                final_level = ai_level
                print(f" Complex emotional content - trusting AI analysis: {ai_level.value}")
            else:
                # Take average if pattern is higher
                avg_index = (ai_index + pattern_index) // 2
                final_level = crisis_levels[min(avg_index, len(crisis_levels) - 1)]
                print(f" Averaging AI and pattern for complex content: {final_level.value}")
        else:
            # For simple content, use safety-first approach (higher level)
            final_level = crisis_levels[max(ai_index, pattern_index)]
            print(f" Safety-first approach for simple content: {final_level.value}")

        # Intelligent harm type reconciliation
        if ai_harm_type == HarmType.BOTH or pattern_harm_type == HarmType.BOTH:
            final_harm_type = HarmType.BOTH
        elif ai_harm_type == HarmType.HARM_TO_OTHERS or pattern_harm_type == HarmType.HARM_TO_OTHERS:
            final_harm_type = HarmType.HARM_TO_OTHERS
        elif ai_harm_type == HarmType.SELF_HARM or pattern_harm_type == HarmType.SELF_HARM:
            final_harm_type = HarmType.SELF_HARM
        else:
            final_harm_type = HarmType.NONE

        return final_level, final_harm_type
```

Figure A.3: Hybrid Reconciliation Logic

```python
class TherapyBot:

    def _get_or_create_session_memory(self, user_id: str, session_id: str) -> SessionMemory:
        """ Get or create session memory with strict isolation, loading from MongoDB if history exists"""
        session_key = f"{user_id}_{session_id}"
        if session_key not in self.session_memories:
            # Try to load existing history from MongoDB to populate memory
            try:
                history = self.storage.get_conversation_history(user_id, session_id, limit=50, skip_token_validation=True)

                if history and len(history) > 0:
                    print(f" Found {len(history)} existing conversations, populating session memory from history")

                    # Extract primary issue from first conversation
                    primary_issue = ""
                    issue_details = {}
                    progress_notes = []
                    key_themes = []

                    # Get primary issue from first meaningful conversation
                    for conv in history[:5]:  # Check first 5 conversations
                        if conv.get('user_input') and len(conv.get('user_input', '')) > 10:
                            # Try to extract issue from user input
                            user_input = conv.get('user_input', '')
                            if not primary_issue and self.llm:
                                try:
                                    issue_prompt = f"""Identify the primary therapeutic concern from this user input: "{user_input[:200]}"

Return ONLY one of these categories:
- work_stress
- relationship_issues
- anxiety_disorders
```

Figure A.4: Session Isolation and Memory Management

Section A.2: Gemini Integration Service

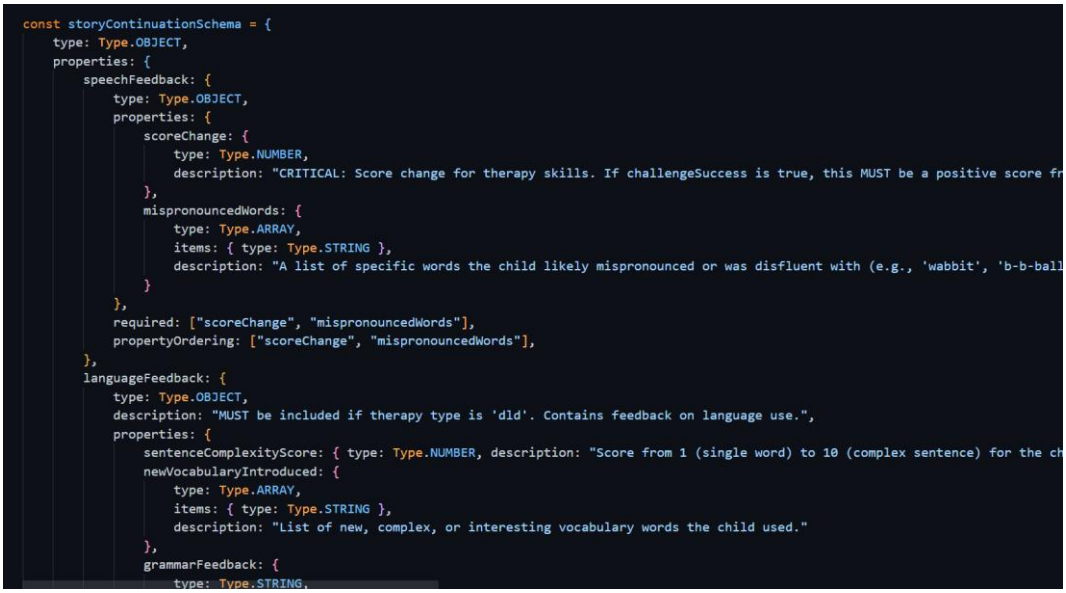


```
export const assessSpeechLevel = async (
    const prompt = `

    --- ANALYSIS WORKFLOW ---
    1.  **Carefully review all three rounds** of the child's assessment data below. Look at each transcript and compare it t

    2.  **For 'pronunciation'**:
        - Focus *specifically* on the clarity of the 'targetWord' within each 'transcript'
        - Compare the child's pronunciation to the expected target word
        - Look for substitutions, omissions, or distortions in the target sounds
        - The 'sentence' provides context but your primary focus is the 'targetWord'

    3.  **For 'fluency'**:
        - Analyze the entire 'transcript' for smoothness and flow
        - Look for repetitions (e.g., "b-b-ball"), prolongations (e.g., "ssssun"), or blocks
        - Count disfluencies and assess their impact on communication
        - Consider the length and complexity of what they were asked to say

    4.  **For 'dld'**:
        - Analyze the entire 'transcript' for language skills
        - Evaluate grammar, sentence structure, vocabulary, and complexity
        - Check if they used complete sentences or just words/phrases
        - Assess their ability to express ideas clearly

    5.  **Assign a precise level** (1-20) based on your holistic analysis:
        - Do NOT default to Level 5 unless the child truly demonstrates moderate skills
        - Be clinically accurate: if they show beginner skills, assign Level 1-5
        - If they show advanced skills, assign Level 16-20
        - Consider the severity and frequency of errors across all 3 rounds
        - Do not simply average the rounds - look at the overall pattern

    6.  **Write clinical reasoning** that explains:
```


Figure A.5: Assessment Prompt Engineering

```
export const continueStory = async (
    const prompt = `
            - -5 points when child fails a challenge
            - 0 points for all regular story-building turns
            - This score reflects performance on therapeutic challenges
    3.  **Level Progression (APPLIES TO ALL THERAPY TYPES)**:
        *   Level increases ONLY after the child successfully completes ALL 5 challenges
        *   This applies to pronunciation, fluency, DLD, and social communication therapy types
        *   When level increases, child earns a unique badge and gets a happy ending
    4.  **Strict Turn Flow**: The game has two modes. After you evaluate a challenge, the VERY NEXT turn MUST be a regular story turn wher
    5.  **CRITICAL CHALLENGE TIMING RULES**:
        *   **ABSOLUTE RULE**: If regularTurnsSinceLastChallenge is 0, you are FORBIDDEN from creating any challenge. You MUST continue th
        *   **MANDATORY CHALLENGE**: If regularTurnsSinceLastChallenge is 2 or more, you MUST create a challenge (unless story is ending).
        *   **OPTIONAL CHALLENGE**: If regularTurnsSinceLastChallenge is 1, you MAY create a challenge or continue the story.
        *   **FIRST CHALLENGE**: If no challenge has been introduced yet, you MUST introduce one now (unless story is ending).
        *   You should NOT introduce challenges too frequently (not every turn)
        *   You should NOT wait too long between challenges (aim for 1-2 regular turns between challenges)
        *   The goal is to balance story progression with therapeutic practice

    --- CURRENT GAME STATE ---
    - Therapy Focus: ${therapyType}
    - Child's Skill Level: ${level} / 20
    - Focus Stars: ${focusStars} / ${MAX_FOCUS_STARS}
    - Challenges Completed This Level: ${speechChallengesCompleted} / ${CHALLENGES_PER_LEVEL}
    - Child Turns So Far: ${childTurnCount}
    - Regular Story Turns Since Last Challenge: ${regularTurnsSinceLastChallenge} (use this to decide if it's time for a new challenge - a
    - Has Any Challenge Been Introduced Yet: ${hasAnyChallengeBeenIntroduced ? 'Yes' : 'No'} (if No, you MUST introduce the first challeng
    - The child just said: "${userInput}"
    - **Was the previous turn a challenge? ${wasPreviousTurnChallenge}**  <-- THIS DETERMINES YOUR MODE.
    - Previous Challenge Details: ${wasPreviousTurnChallenge ? JSON.stringify(lastChallenge) : 'N/A'}

    --- YOUR TASK: CHOOSE A MODE AND FOLLOW ITS RULES EXACTLY ---
```

Figure A.6: Story Game Logic & Challenge Timing



```
const storyContinuationSchema = {
    type: Type.OBJECT,
    properties: {
        speechFeedback: {
            type: Type.OBJECT,
            properties: {
                scoreChange: {
                    type: Type.NUMBER,
                    description: "CRITICAL: Score change for therapy skills. If challengeSuccess is true, this MUST be a positive score fr
                },
                mispronouncedWords: {
                    type: Type.ARRAY,
                    items: { type: Type.STRING },
                    description: "A list of specific words the child likely mispronounced or was disfluent with (e.g., 'wabbit', 'b-b-ball
                }
            },
            required: ["scoreChange", "mispronouncedWords"],
            propertyOrdering: ["scoreChange", "mispronouncedWords"],
        },
        languageFeedback: {
            type: Type.OBJECT,
            description: "MUST be included if therapy type is 'dld'. Contains feedback on language use.",
            properties: {
                sentenceComplexityScore: { type: Type.NUMBER, description: "Score from 1 (single word) to 10 (complex sentence) for the ch
                newVocabularyIntroduced: {
                    type: Type.ARRAY,
                    items: { type: Type.STRING },
                    description: "List of new, complex, or interesting vocabulary words the child used."
                },
                grammarFeedback: {
                    type: Type.STRING,
```

Figure A.7: Structured JSON Output Schema

```
async function retryApiCall<T>(
    apiCall: (ai: GoogleGenAI) => Promise<T>, // Takes the AI client as an argument
    maxRetries: number = 3,
    initialDelay: number = 1000
): Promise<T> {
    let lastError: Error | null = null;

    for (let attempt = 0; attempt < maxRetries; attempt++) {
        try {
            if (!apiKey) {
                throw new Error("An API Key must be set when running in a browser. Please set VITE_GEMINI_API_KEY in your .env file.");
            }
            const ai = new GoogleGenAI({ apiKey: apiKey }); // Re-initialize for fresh API key
            return await apiCall(ai);
        } catch (error: any) {
            lastError = error;
            const errorMessage = error?.message || error?.toString() || '';
            const errorCode = error?.code || error?.status;

            // Check for API key authentication errors first - don't retry
            if (errorCode === 401 || errorCode === 403 || errorMessage.includes('API key') || errorMessage.includes('authentication') || e
                throw new Error(`API key authentication failed. Please verify your API_KEY environment variable. Error: ${errorMessage}`);
            }

            // Check for invalid model name (404) - don't retry, throw immediately with helpful message
            if (errorCode === 404 || errorMessage.includes('NOT_FOUND') || errorMessage.includes('model') && errorMessage.includes('not fo
                const modelInError = errorMessage.match(/model\/([^ ]+)/)?.[1] || 'the specified model';
                throw new Error(`Invalid model name: ${modelInError}. Please check the model name in geminiService.ts and verify your API
            }

            // Check for quota exceeded (429) - extract retry delay
```

Figure A.8: API Retry Mechanism & Error Handling

Plagiarism Report

FYP

ORIGINALITY REPORT

6%	6%	4%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	1%
2	arxiv.org Internet Source	<1%
3	Sahaya Jesto, Bijo Kunnumpurath. "The future of digital mental health care: challenges and opportunities for practice", Elsevier BV, 2025 Publication	<1%
4	sajip.co.za Internet Source	<1%
5	ejournal.uin-suka.ac.id Internet Source	<1%
6	www.preprints.org Internet Source	<1%
7	journal.arteii.or.id Internet Source	<1%
8	etd.lib.metu.edu.tr Internet Source	<1%
9	trepo.tuni.fi Internet Source	<1%
10	thomasmore.be Internet Source	<1%
11	www.omicsonline.org Internet Source	<1%
12	ir.uitm.edu.my Internet Source	<1%

13	www.frontiersin.org Internet Source	<1%
14	assets-eu.researchsquare.com Internet Source	<1%
15	digitalcollection.utem.edu.my Internet Source	<1%
16	huggingface.co Internet Source	<1%
17	journals.stmjournals.com Internet Source	<1%
18	edtechbooks.org Internet Source	<1%
19	repository.seku.ac.ke Internet Source	<1%
20	open.uct.ac.za Internet Source	<1%
21	www.doria.fi Internet Source	<1%
22	pmc.ncbi.nlm.nih.gov Internet Source	<1%
23	Duran, Mustafa Berk. "Comparative Evaluation of Command Distribution via DDS and CORBA in a Software Reference Architecture.", Middle East Technical University (Turkey), 2024 Publication	<1%
24	www.mextesol.net Internet Source	<1%
25	ailbry.com Internet Source	<1%
	ir.lib.uth.gr	

26	Internet Source	<1%
27	aplusgardencentre.weebly.com Internet Source	<1%
28	Robbert Sanderman, Karen Morgan. "The Routledge International Handbook of Health Psychology - Global and Contemporary Issues", Routledge, 2025 Publication	<1%
29	Shalli Rani, Ayush Dogra, Ashu Taneja. "Smart Computing and Communication for Sustainable Convergence", CRC Press, 2025 Publication	<1%
30	Veronika Nugraheni Sri Lestari, Dwi Cahyono, Sri Susilowati. "Designing Early Warning System for The Impact of Industrial Development Waste by Using pH Control System", Open Science Framework, 2018 Publication	<1%
31	academica-e.unavarra.es Internet Source	<1%
32	cuir.car.chula.ac.th Internet Source	<1%
33	fit.repo.nii.ac.jp Internet Source	<1%
34	repozitorij.uni-lj.si Internet Source	<1%
35	ses.library.usyd.edu.au Internet Source	<1%
36	www.br.freelancer.com Internet Source	<1%
37	www.dermatoglyphics.com Internet Source	<1%
38	"Text, Speech, and Dialogue", Springer Science and Business Media LLC, 2026 Publication	<1%
39	Zhuoyang Li, Zihao Zhu, Xinning Gui, Yuhan Luo. ""This is human intelligence debugging artificial intelligence": Examining how people prompt GPT in seeking mental health support", International Journal of Human-Computer Studies, 2025 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		