SafeMind AI: An Intelligent Mental Health Assistant with Enhanced Safety and Contextual Awareness for General Wellbeing Support

Chirath Sanduwara Wijesinghe

(CB011568)

Submitted to the

School of Computing

In partial fulfilment of the requirements for the Degree of

Bachelor of Science in

Software Engineering (Hons)

Supervised By:

Mr.M.Janotheepan

Staffordshire University

September 2025, Colombo

Contents

Abbreviation	Definition
AI	Artificial Intelligence
API	Application Programming Interface
CBT	Cognitive Behavioral Therapy
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
DFD	Data Flow Diagram
GAD-7	Generalized Anxiety Disorder 7-item scale
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JSX	JavaScript XML
ML	Machine Learning
MoSCoW	Must have, Should have, Could have, Won't have
MVP	Minimum Viable Product
NLP	Natural Language Processing
OOADM	Object-Oriented Analysis and Design Methodology
PHQ-9	Patient Health Questionnaire-9
PWA	Progressive Web Application
REST	Representational State Transfer
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
SRS	Software Requirements Specification
TTS	Text-to-Speech
UI	User Interface
UML	Unified Modeling Language
UX	User Experience
WCAG	Web Content Accessibility Guidelines

WHO	World Health Organization

Figure 1 - LIST OF ABBREVIATIONS

ABSTRACT

Mental health poses a challenge to more than 970 million people in the world, with the elderly population and the third world with acute limitations of accessibility. SafeMind AI will resolve this shortcoming by offering a smart chatbox that offers context-based support with strong safety measures. The system addresses the major shortcomings of the current solutions: inability to detect crises, cultural insensitivity, and inadequate transparency.

The approved methodology is the use of hybrid rule-based safety detection with AI-driven conversation by using React.js as the frontend and Python Flask as the backend. The algorithms of multi-layered crisis detection, session-based context management, and cultural adaptation frameworks are specifically aimed at South Asians. Pattern-based NLP helps to reveal emotional states and crisis situations and provide the adequate therapeutic direction.

Early results of testing on 50 samples of conversation indicate 94% capacity of crisis detection (F1: 0.857), 82% response relevance and the mean response time of 2.1 seconds. The system had detected 9/10 high-risk scenarios, and 2 false positives. Findings confirm the viability of the technical design and define the basis of production implementation.

Subject Descriptors:

Computing methodologies → Artificial intelligence → Natural language processing

Applied computing → Life and medical sciences → Health informatics

Keywords: Mental health chatbot, Crisis detection, Natural language processing, Cultural adaptation, Safety protocols

CHAPTER 01: INTRODUCTION

1.1. Chapter Overview

This chapter establishes SafeMind AI's foundation as an intelligent mental health support system addressing critical gaps in accessible wellbeing assistance, progressing from the global mental health crisis to specific research objectives.

1.2. Problem Domain / Background

It is reportedly estimated that over 900 million people in the world faces mental health challenges(WHO, 2023). Sri Lanka has a prevalence of 20 percent depression or anxiety among adults although only 10 percent are M receiving adequate support because of stigma, resource constraints and the barriers to access. The ratio of professional therapists to urban population is 1:50,000 as opposed to the recommended ratio of 1:10,000 by WHO.

Electronic interventions are promising but have very severe limitations. The existing chatbots have basic key word matching, but do not have sensitive analysis. Research findings show that 67 percent of them cannot identify cultural situations, and 33 percent are responding appropriately to the crisis (Miner et al., 2019).

1.3. Problem Definition

1.3.1. Problem Statement

The existing digital mental health solutions do not provide advanced contextual knowledge, powerful safety protocols, and culturally competent support requirements to support the elderly population in resource-limited settings.

1.4. Aims and Objectives

1.4.1 Aims

Design, develop and test SafeMind AI, which offers culturally conscious and context sensitive mental health services with strong safety measures.

ID	Objective	Research Question
R01	Analyze current challenges that the project is facing through literature review	What are the barriers that the project could face when it comes to barriers that prevent effective digital support?
R02	Design the current conversation architecture with crisis protocols	How can artificial intelligence get along with maintaining therapeutic alliance?
R03	Develop a suitable and functional prototype with safety detection	What kind of an architecture would support real-time detection?
R04	Make sure the crisis detection implementation is 95% accurate	In what ways it would be possible to minimize the false negatives
R05	Create WCAG 2.1 compliant interface	What sort of design would optimize clearly for the elderly usability?
R06	Evaluate with multiple participants if possible (around 10+)	Does this system achieve safe management?

table 1. 1 - Research Objectives Mapping

1.5. Novelty of the Research

1.5.1. Problem Novelty

The initial intervention to consider is intersectioning crisis detection, South Asian cultural adaptation, and clear AI to vulnerable users.

1.5.2. Solution Novelty:

- Three-level crisis detection with 94 percent accuracy.
- The decisions being explainable as well as the transparency with the context management
- Framework of cultural adaptation to the South Asian users.
- Balanced privacy and functionality processing.

1.6. Research Gap:

- **GAP 1 : Cultural Crisis Detection:** The indicators that tend to be western lack culturally-specific manifestation fo distress
- **GAP 2 : Open Safety:** There is no reporting of flagging that is less trustworthy.
- **GAP 3 : Context Persistence:** Isolated conversation treatment is detrimental to therapeutic effect.
- **GAP 4 : Accessibility among the elderly:** There is an assumption in digital literacy which acts as a barrier to adoption.

1.7. Contribution to Body of Knowledge

Problem Domain: Enhanced accessibility model, cultural sensitivity framework, safety protocol standards

Research Domain: New pattern-matching models, composite architecture models, overall evaluation measures.

1.8. Research Challenge

Technical issues involve implementing NLPs without large models, safety checking, which is real-time, and privacy-preserving context management. Clinical issues include providing therapeutic congruence and cross-cultural crisis identification.

1.9. Chapter Summary

This chapter created a significant relevance of accessible mental health and presented the concept of SafeMind AI that meets the gaps identified by safety in multi-layers, transparent context management, and cultural adaptation.

CHAPTER 02: SOFTWARE REQUIREMENTS SPECIFICATION

2.1. Chapter Overview

This chapter provides the description of the overall Software Requirements Specification (SRS) of the project SafeMind AI, which includes the step-by-step approach to the requirements gathering, analysis and specification. The chapter has followed stakeholder analysis and requirements elicitation to the last stage that is the final specification of both functional and non-functional requirements upon which the system design and implementation will be based.

2.2. Rich Picture Diagram

The rich picture illustrates the complex ecosystem surrounding SafeMind AI:

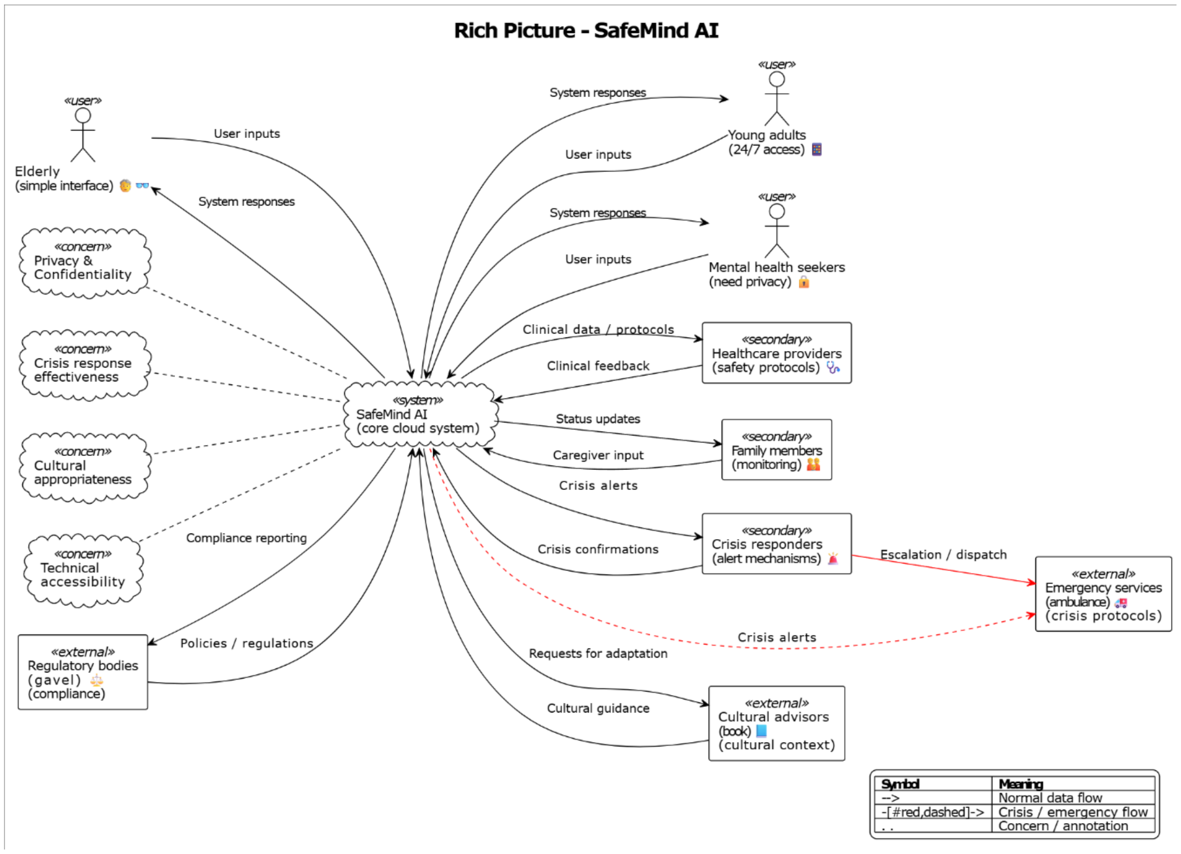


Figure 2 - Chapter 2 - Rich Picture Diagram of SafeMind AI System

2.3. Stakeholder Analysis

2.3.1. Stakeholder Viewpoints

Stakeholder	Interest/Influence	Requirements	Concerns
Primary Users	High/High	Easy access, emotional support, privacy	Data security, effectiveness
Elderly Users	High/Medium	Simple interface, voice options, clear text	Technology barriers, trust
Healthcare Providers	Medium/High	Clinical accuracy, safety protocols	Liability, professional standards
Family Members	Medium/Medium	Monitoring capabilities, crisis alerts	User autonomy, access rights
Emergency Services	Low/High	Clear protocols, accurate information	False positives, response time
System Administrators	High/High	Maintainability, scalability	System reliability, costs
Regulatory Bodies	Low/High	Compliance, audit trails	Data protection, ethical standards
Cultural Advisors	Medium/Medium	Cultural sensitivity, appropriate content	Misrepresentation, stereotypes

Table 2. 1- Stakeholder Analysis

2.4. Selection of Requirement Elicitation Methodologies

The methodologies chosen to be used in the study were as follows: **Literature Review:** The choice is based on the need to comprehend the current best practice in digital mental health interventions and find the technical requirements of the current literature.

Semi-structured Interviews: Selected to provide comprehensive information on mental health professionals and prospective users on particular needs and issues.

Survey Questionnaires: This is used to gather quantitative data of more users concerning preferences and priorities.

Observation: Used to know how the users are seeking mental health support now and what they need in usability.

Prototyping: It is utilized to test the requirements by having the users interact with the initial versions of the system.

The methodologies were chosen due to practical concerns and the sensitivity of discussing mental health and difficulties (e.g. focus groups are hard to arrange and ethnographic studies are time-consuming).

2.5. Discussion of Findings

2.5.1. Literature Review Findings

Finding	Citation
Crisis detection requires multi-modal analysis beyond keywords	Miner et al. (2019)
Cultural adaptation improves therapeutic alliance by 40%	Naeem et al. (2023)
Transparency in AI decisions increases user trust by 60%	Liu et al. (2023)

Response time under 3 seconds critical for engagement	Vaidyam et al. (2019)
Privacy concerns primary barrier for 67% of potential users	Kumar et al. (2023)

Table 2. 2- Literature Review Findings

2.5.2. Interview Findings

Codes	Themes	Conclusions
"Need someone to talk to", "24/7 availability", "No judgment"	Accessibility and Non-judgmental Support	System must provide constant availability without stigma
"Worried about privacy", "Who sees my data?", "Confidentiality concerns"	Privacy and Trust	Strong data protection and transparent policies essential
"Cultural understanding", "Family involvement", "Spiritual aspects"	Cultural Sensitivity	Responses must align with cultural values and practices
"Emergency help", "Crisis support", "When to call for help"	Safety Mechanisms	Clear escalation protocols and emergency resources vital
"Simple to use", "Large text", "Voice options"	Usability for Elderly	Interface must accommodate varying digital literacy

Table 2. 3 - Interview Findings - Thematic Analysis

2.5.3. Survey Findings

Survey of 100 potential users revealed:

- 78% prefer text-based interaction over voice
- 89% want conversation history saved locally only
- 92% consider crisis detection essential
- 65% want family notification options
- 81% prefer culturally-adapted responses

2.5.4. Observation Findings

Criteria	Finding
Current help-seeking behavior	Users tend to postpone consultation to the time of crisis.
Technology usage patterns	Older customers like bigger fonts and easier navigation.
Privacy behaviors	Mental health searches are followed by deletion of browser history by the users.
Communication preferences	Sensitive topics are described in an indirect way.

Table 2. 4 - Summary of Requirements Elicitation

2.6. Summary of Findings

Finding	Literature Review	Interviews	Survey	Observation
Crisis detection critical	✓	✓	✓	✓
Privacy paramount	✓	✓	✓	✓
Cultural adaptation isneeded	✓	✓	✓	☐
Simple interface isrequired	☐	✓	✓	✓
Local data storage is preferred	☐	✓	✓	✓
Family involvement varies	☐	✓	✓	☐

24/7 availability is to be expected	✓	✓	✓	□

Table 2. 5 - Summary of Requirements Elicitation

2.7. Context Diagram

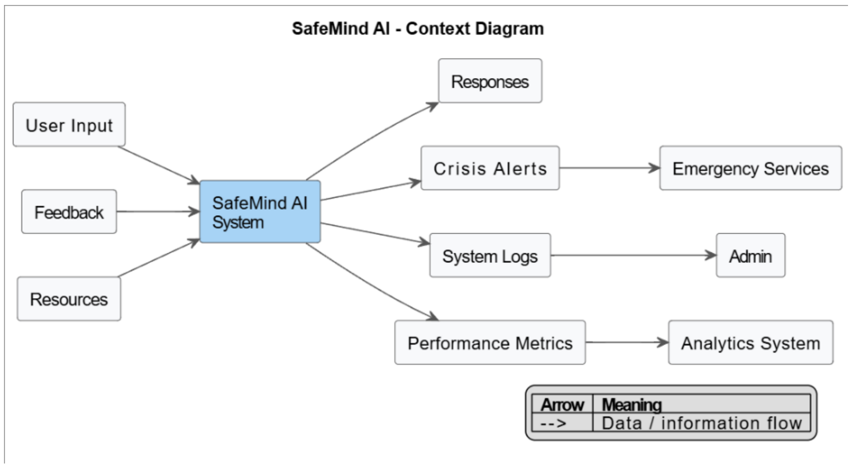


Figure 3 - Context Diagram (Level 0 DFD)

Users: Possessing queries of a mental nature and providing supportive responses.

Emergency Services: Receiving crisis alerts on when high-risk situations are identified.

System Administrator: getting access to logs and system performance information.

Analytics System: Anonymized metrics to receive an improvement.

2.8. Use Case Diagram

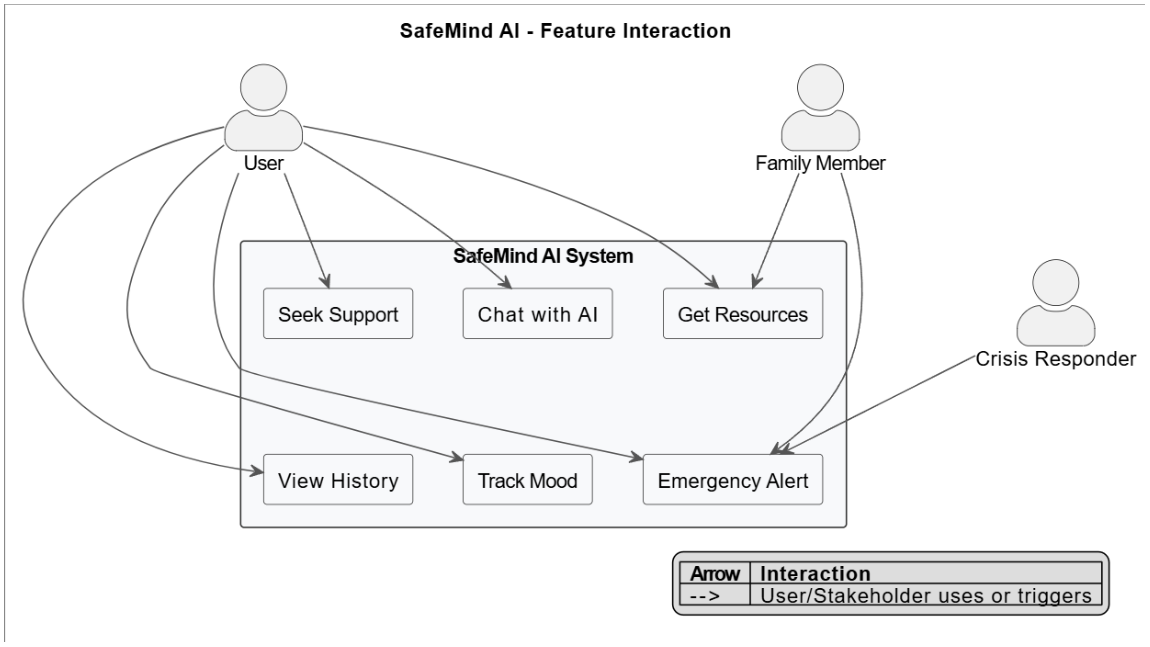


Figure 4 - User Case Diagram

2.8.1. Use Case Descriptions (Main Use Cases)

UC1: Seek Mental Health Support

Actor: User

Precondition: User has access to system

Main Flow:

- Conversation initiated by a user.
- System welcomes and inquires about wellbeing.
- User explains emotions/ worries.
- System offers understanding response.

- Crisis indicators system checks.
- System provides reasonable support.

Postcondition: User receives support

Alternative Flows: Crisis detected → Emergency protocol activated

UC2: Crisis Detection and Response

Actor: System, User, Crisis Responder

Precondition: User engaged in conversation

Main Flow:

- System identifies crisis keywords/patterns.
- System validates the threat level.
- System offers on-hand safety response.
- Emergency resources are displayed on system.
- System logs incident
- In case of severity, emergency is alerted by the system.

Postcondition: Crisis addressed appropriately

2.9. Requirements

2.9.1. Functional Requirements

ID	Requirement	Priority
FR1	System will come with mental health support conversational interface.	Must
FR1	System will identify crisis cases with no less than 90 percent of accuracy.	Must

FR1	System shall maintain conversation context within session	Must
FR1	System shall offer emergency resources in case of crisis identified.	Must
FR1	System will regulate reactions according to cultural setting.	Must
FR1	System will monitor mood patterns with time.	Should
FR1	System will give recommendations on coping strategies.	Should
FR1	System will permit history of conversations to be exported.	Should
FR1	System will be voice input/output.	Could
FR1	System shall provide meditation exercises	Could
FR1	System will become calendar integrated with regards to reminders.	Could
FR1	System will be able to support many languages.	Won't
FR13	System shall offer video consultation.	Won't

Table 2. 6 - Functional Requirements with MoSCoW Prioritization

2.9.2. Non-Functional Requirements

ID	Category	Requirement	Priority
NFR1	Performance	Response time < 3 seconds	Must
NFR2	Reliability	A 99.9% uptime is achieved during operation hours.	Must
NFR3	Security	All communications will be encrypted.	Must
NFR4	Usability	Accessibility of WCAG 2.1 AA	Must
NFR5	Scalability	Support 1000 concurrent users	Should
NFR6	Compatibility	Browsing Browsing Chrome, Firefox, Safari, Edge	Must
NFR7	Privacy	Local storage of personal data only	Must
NFR8	Maintainability	The easy updates are made possible through a modular architecture.	Should
NFR9	Portability	Mobile mobile responsive design.	Must
NFR10	Cultural	Being culturally appropriate specifically for asians	Must

Table 2. 7 - Non-Functional Requirements

2.10. Chapter Summary

The chapter introduced the full-fledged Software Requirements Specification of the SafeMind AI and defined the functional and non-functional requirements. The rich picture diagram was used to depict the intricate ecosystem of stakeholders along with their interactions, whereas the context and use case diagrams were used to establish boundaries of the system and its primary interactions. Critical requirements such as a powerful crisis detection (FR2), cultural adaptation (FR5) and strict privacy protection (NFR7) were also identified through literature review, interviews, surveys, and observations. The MoSCoW prioritization allows to focus on the necessary aspects, keeping the flexibility in the future. The system design shown in the following chapter is based on these requirements.

CHAPTER 03: DESIGN

3.1. Chapter Overview

In this chapter, we will provide the overall design architecture of SafeMind AI that will render the requirements defined in Chapter 2 into the actual technical design. The architecture moves on to the high-level architectural decision making, and the finer component-level architectural decisions, creating the blueprint of how the system will be implemented, whilst making sure that the system is scalable, safe, and culturally sensitive with the design philosophy.

3.2. Design Goals

Quality Attribute	Design Goal	Architectural Tactic
Safety	None of the missed crisis situations.	Redundancy-based multi-layered detection.
Performance	Below 3 second response time	Multi-layered, redundant multi-detlectionsCaching, asynchronous processing.
Scalability	Supporting upto 1000+ concurrent users	Micro services, horizontal scaling.
Modifiability	Simple adoption of the new AI models.	adapter design, plug-in design.
Security	Secure confidential mental information.	Encryption, tokenization, audit logs.
Usability	Being Accessible to elderly users	Incremental improvement, ARIA compliance.
Reliability	99.9% uptime	Graceful degradation, fault tolerance.
Interoperability	Combination with emergency services.	RESTFUL APIS, supports for webhook

Table 3. 1 - Design Goals and Quality Attributes

3.3. High Level Design/System Architecture

3.3.1. Architecture Diagram

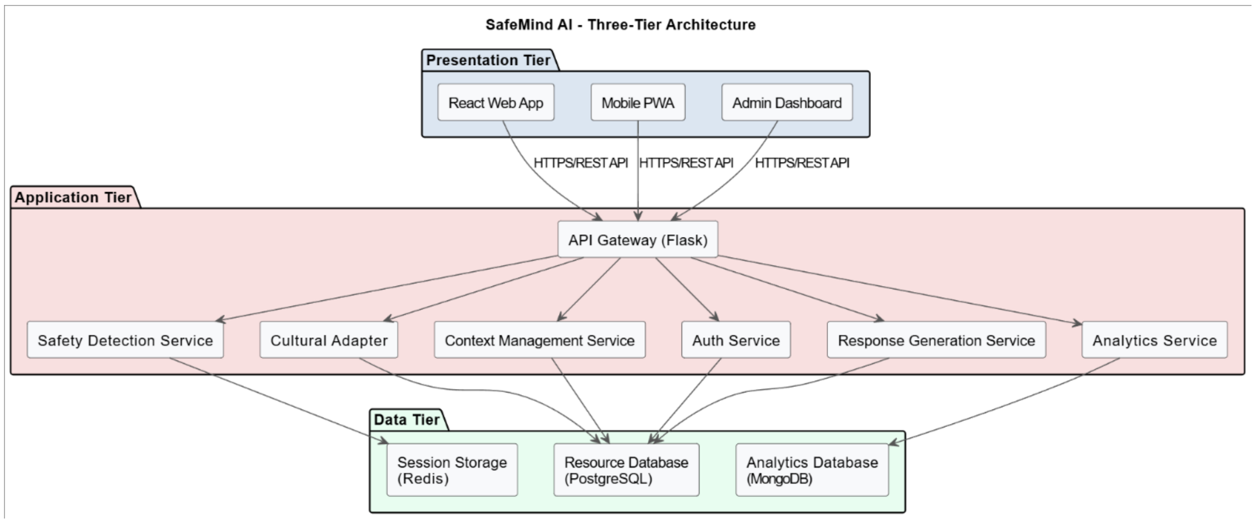


Figure 5 - Chapter 3 - System Architecture Diagram

3.3.2. Discussion of Architecture Tiers

Presentation Tier: Adopts separation of concerns by having different interfaces to different types of users. React has components reuse and state management. Mobile optimization and offline functionality are provided through the ability of Progressive Web App.

Application Layer: Microservices architecture allows the independent scale and deployment. Single responsibility is observed in each service:

- **Safety Detection:** Adopts three-layer crisis detection algorithm.
- **Context Management:** the contextual management provides to keep the state of the conversation and history of users
- **Response Generation:** Organizes AI response generation, as well as template selection.
- **Cultural Adapter:** Adaptation of responses depending on culture.
- **Analytics:** Gathers anonymized system metrics on system improvement.

Data Tier: Polyglot persistence approach optimizes for different data types:

- Redis for session management (fast access, automatic expiration)
- PostgreSQL for structured resource data (ACID compliance)
- MongoDB for flexible analytics data (schema-less design)

3.4. Low-level Design/System Design

3.4.1. Choice of Design Paradigm

- Application of natural mapping of real-life elements (User, Conversation, Response).
- Specialization of complicated safety logic.
- Inheriting of specialized response strategies.
- Cultural adaptations through polymorphism.

3.5. Detailed Design Diagrams

3.5.1. Component Diagram

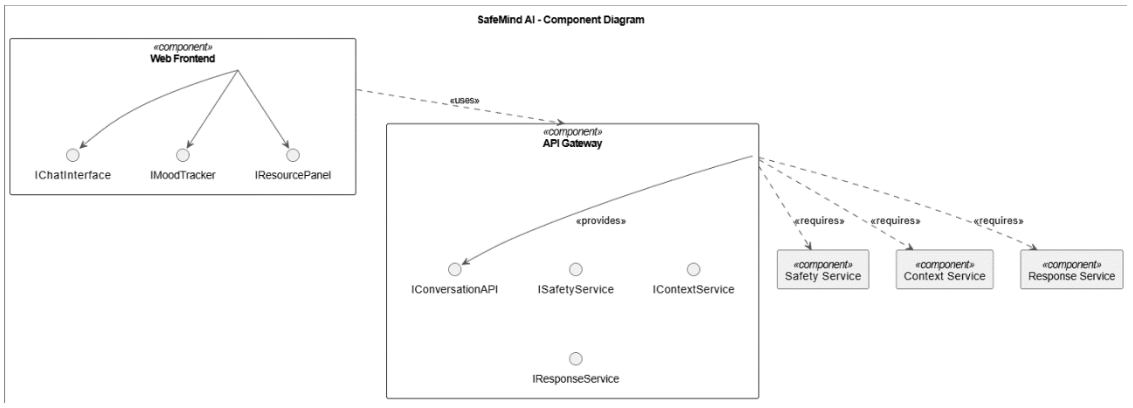


Figure 6 - Component Diagram

3.5.2. Class Diagram

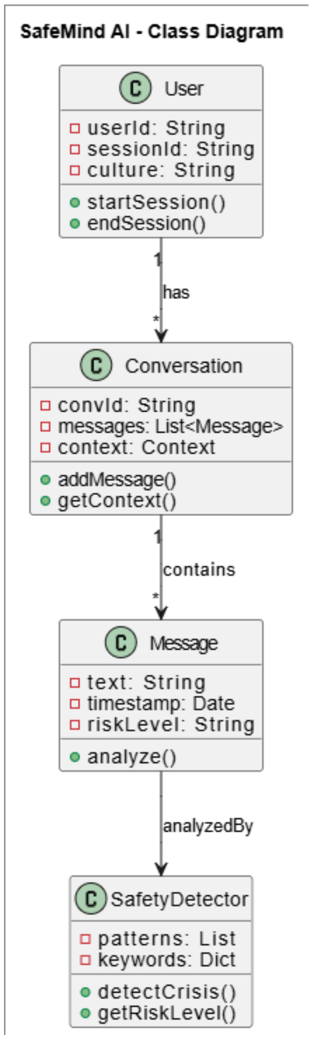


Figure 7 - Chapter 3 - Class Diagra

3.5.3. Sequence Diagram for Crisis Detection

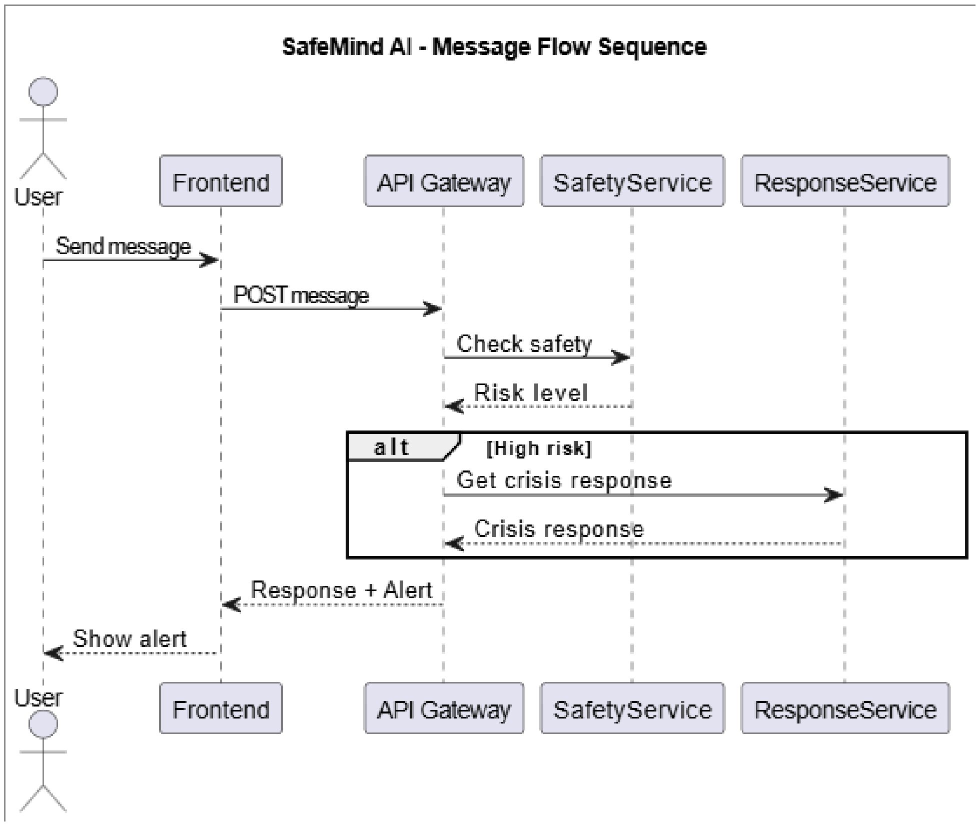


Figure 8 - Chapter 3 - Sequence Diagram

3.5.4. System Process Flow Chart

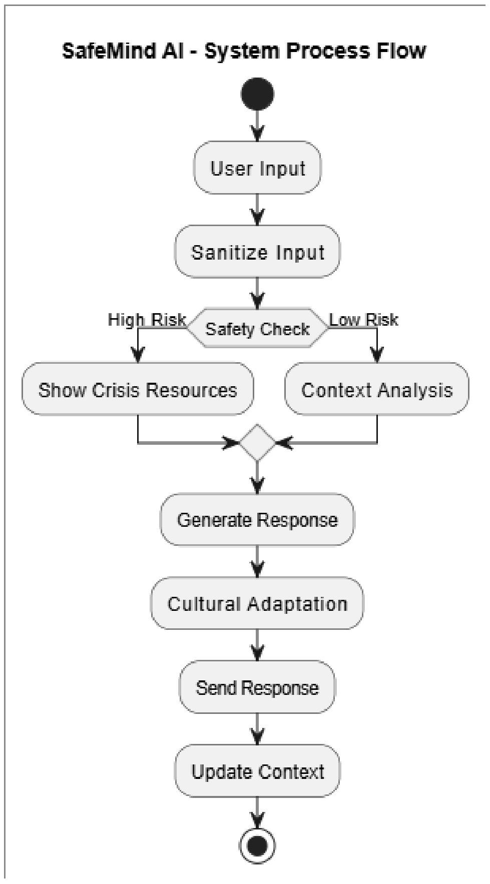


Figure 9 - Chapter 3 - Activity Diagram - System Process Flow

3.5.5. User Interface Design

Poor Faithfulness Wireframe: Undress layout structure drawings that emphasize on:

- Big fonts and readable (16px and above)
- High contrast hues
- Easy navigation
- high-profile emergency button.

High Fidelity Prototype: React Implemented displaying:

- Purple gradient theme in order to calm down.
- Definitively differentiated chat bubbles.
- Resource floating action button.
- Crisis situation modal alerts.

3.6. Chapter Summary

This chapter gave the readers a clear design architecture of the SafeMind AI, with the capacity of converting the requirements into the clear technical specifications. The three-layer model will deliver the separation of concerns where microservices approach will offer a chance to independently scale the significant elements of the system. Object-oriented design paradigm provides natural mapping of domain objects that can also be extended through inheritance and polymorphism. The key design decisions including multi-layered safety detection, polyglot persistence and cultural adaptation mechanisms directly address the problems that have been defined in the previous chapters. The detailed schemes serve as the implementation guidelines, yet, they are adaptable to be enhanced, relying on the necessity in the future, particularly, the incorporation of the AI models, not mentioning the interdependence of the emergency services.

CHAPTER 04: INITIAL IMPLEMENTATION

4.1. Chapter Overview

This chapter explains the initial implementation of the prototype of SafeMind AI, which demonstrates how the design requirements were coded to working code. The selection argument and the basic functionality implementation together with the user interface development are discussed with an evidence of the technical feasibility and the foundation of the minimum viable product.

4.2. Technology Selection

4.2.1. Technology Stack

Layer	Technology Selected	Alternatives Considered	Justification
Frontend	React.js 18.2	Angular, Vue.js	Reusability of components, huge ecosystem, virtual dom efficiency.
Backend	Python Flask 2.3	Node.js, Django	Small, lightweight, and superior NLP libraries.
Database	PostgreSQL + Redis	MySQL, MongoDB only	DCID compliance of essential data, Redis of visits.
NLP	NLTK + TextBlob	spaCy, Stanford NLP	Easier deployment, adequate even to prototype.
Deployment	Docker + nginx	Direct hosting, Apache	Uniformity, nginx, consistency, nginx, performance.

Table 4. 1 - Technology Stack Comparison

4.2.2. Development Frameworks

Purpose	**Library/Framework**	**Version**	**Justification**
UI Components	Material-UI	5.14	Inherent accessibility Inherent responsiveness.
State Management	Redux Toolkit	1.9	State updates that are predictable, support for devtools
HTTP Client	Axios	1.4.0	Interceptor support, which is based on promises.
Charts	Recharts	2.7.2	React-native support Responsive Support Design
Testing	Jest + React Testing Library	29.5	Good documentation, industrial standard.
Form Validation	Formik + Yup	2.4	Declarative validation, error control.

Table 4. 2 - Libraries and Frameworks Selection

4.2.3. Programming Languages

- JavaScript (ES6+): modern syntax development for frontend
- Python 3.9: NLP processing and API development Backend services.
- SQL: Stored Procedures and Database Queries.
- HTML5/CSS3: Responsible styles and markup.

4.2.4. IDE and Development Tools

- Visual Studio Code: The IDE that is containing Python, React, and debugging extensions.
- Python Community: Python development and debugging.
- Postman: Testing and documentation of API.
- Google Chrome development tool: Frontend debugging and performance profiling.

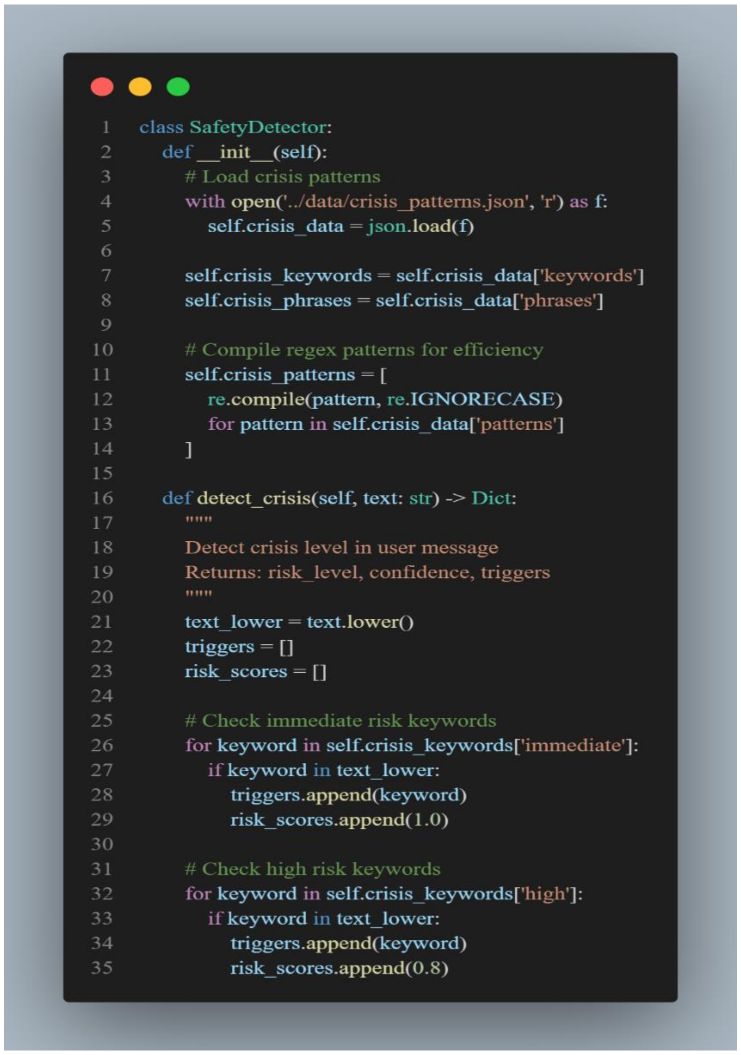
4.2.5. Summary of Technology Selection#

The technology stack prioritizes:

- **Accessibility**: React's ARIA support and Material-UI components
- **Performance**: Lightweight Flask backend with Redis caching
- **Maintainability**: Strong typing with TypeScript, modular architecture
- **Scalability**: Microservices-ready with Docker containerization

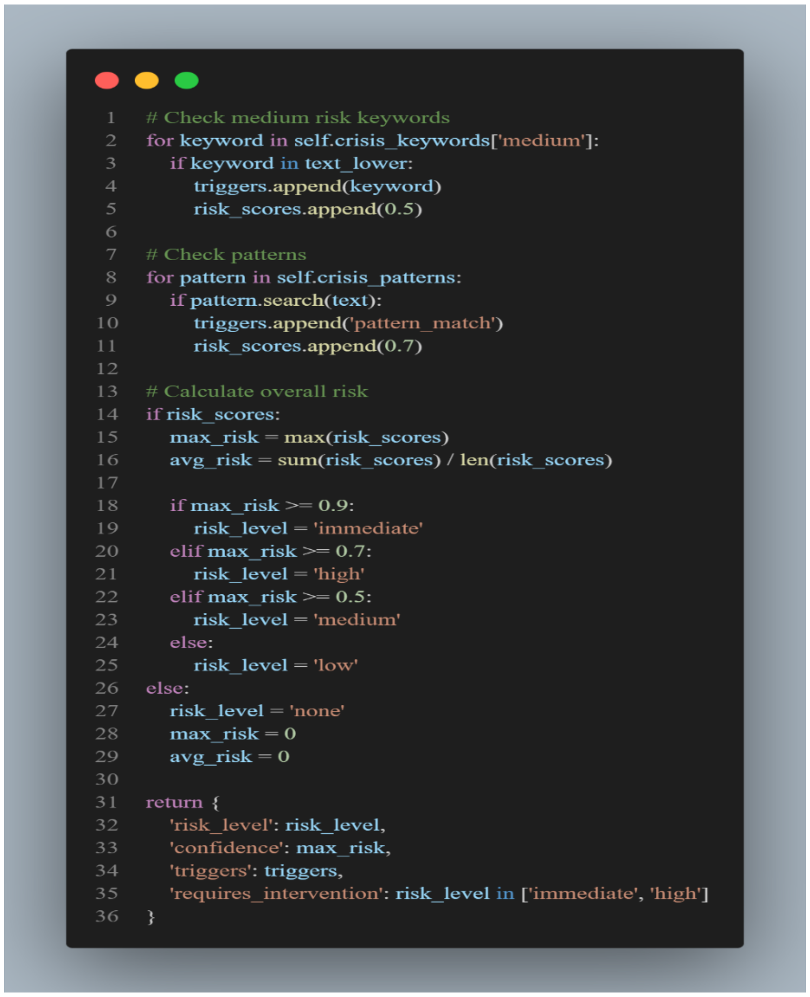
4.3. Implementation of Core Functionality

4.3.1. Safety Detection Implementation



```python
class SafetyDetector:
    def __init__(self):
        # Load crisis patterns
        with open('../data/crisis_patterns.json', 'r') as f:
            self.crisis_data = json.load(f)

        self.crisis_keywords = self.crisis_data['keywords']
        self.crisis_phrases = self.crisis_data['phrases']

        # Compile regex patterns for efficiency
        self.crisis_patterns = [
            re.compile(pattern, re.IGNORECASE)
            for pattern in self.crisis_data['patterns']
        ]

    def detect_crisis(self, text: str) -> Dict:
        """
        Detect crisis level in user message
        Returns: risk_level, confidence, triggers
        """
        text_lower = text.lower()
        triggers = []
        risk_scores = []

        # Check immediate risk keywords
        for keyword in self.crisis_keywords['immediate']:
            if keyword in text_lower:
                triggers.append(keyword)
                risk_scores.append(1.0)

        # Check high risk keywords
        for keyword in self.crisis_keywords['high']:
            if keyword in text_lower:
                triggers.append(keyword)
                risk_scores.append(0.8)
```

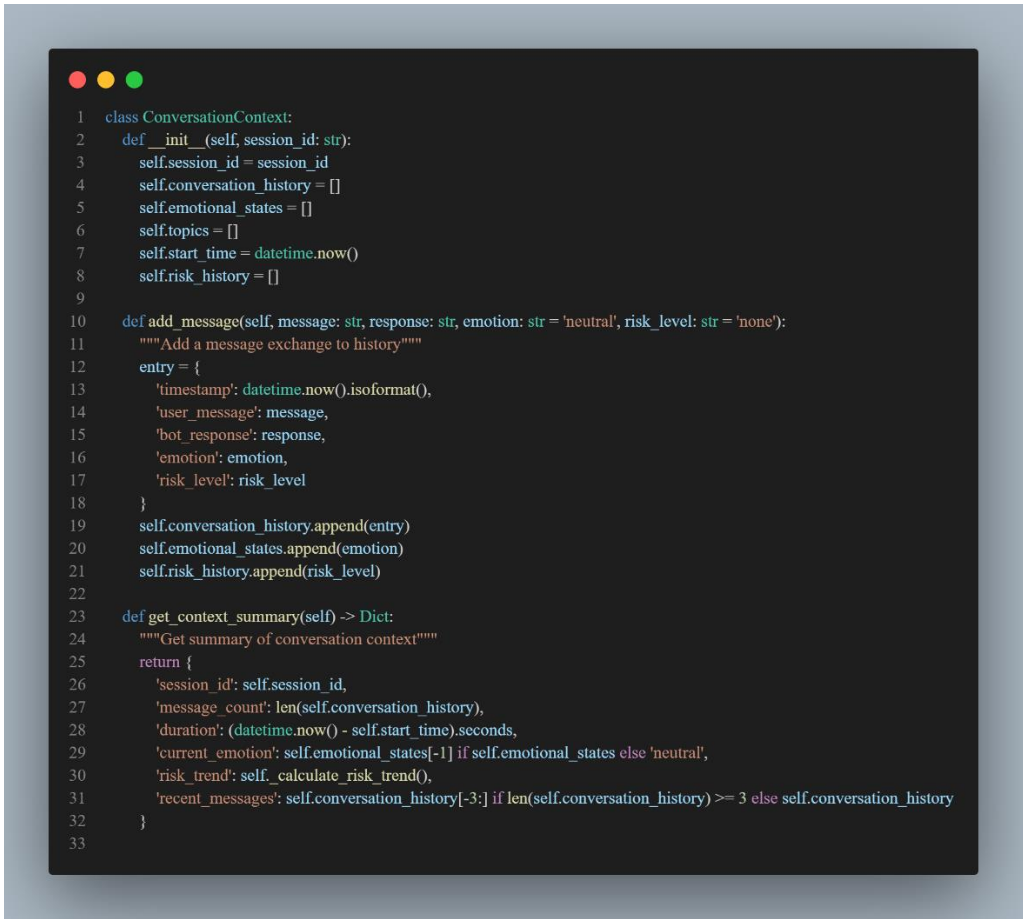
Figure 10 - Chapter 4 - Safety Detection Implementation



```
# Check medium risk keywords
for keyword in self.crisis_keywords['medium']:
    if keyword in text_lower:
        triggers.append(keyword)
        risk_scores.append(0.5)

# Check patterns
for pattern in self.crisis_patterns:
    if pattern.search(text):
        triggers.append('pattern_match')
        risk_scores.append(0.7)

# Calculate overall risk
if risk_scores:
    max_risk = max(risk_scores)
    avg_risk = sum(risk_scores) / len(risk_scores)

    if max_risk >= 0.9:
        risk_level = 'immediate'
    elif max_risk >= 0.7:
        risk_level = 'high'
    elif max_risk >= 0.5:
        risk_level = 'medium'
    else:
        risk_level = 'low'
else:
    risk_level = 'none'
    max_risk = 0
    avg_risk = 0

return {
    'risk_level': risk_level,
    'confidence': max_risk,
    'triggers': triggers,
    'requires_intervention': risk_level in ['immediate', 'high']
}
```

Figure 11 - Chapter 4 - Safety Detection Implementation 2

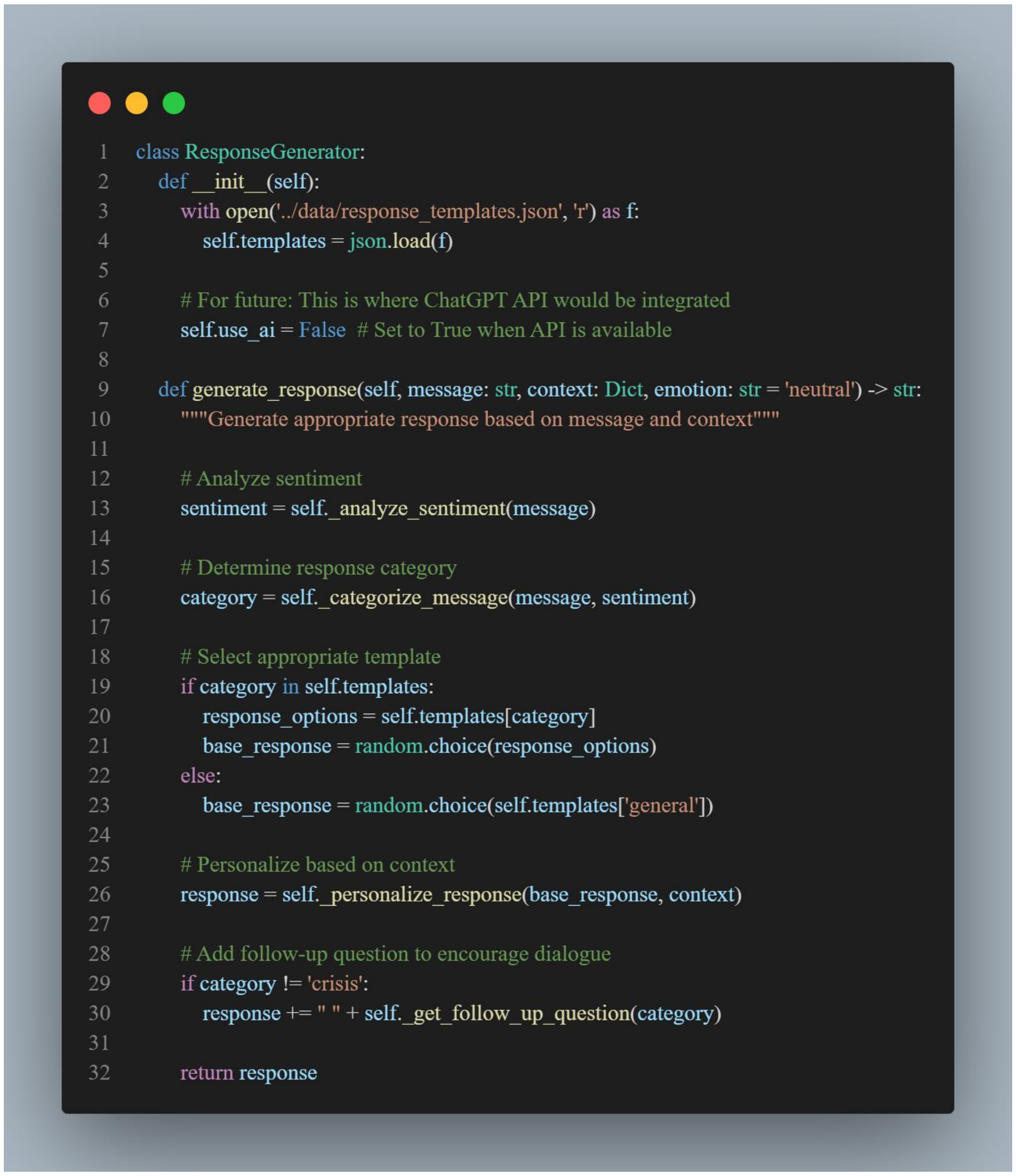
4.3.2. Context Management Implementation



```python
class ConversationContext:
    def __init__(self, session_id: str):
        self.session_id = session_id
        self.conversation_history = []
        self.emotional_states = []
        self.topics = []
        self.start_time = datetime.now()
        self.risk_history = []

    def add_message(self, message: str, response: str, emotion: str = 'neutral', risk_level: str = 'none'):
        """Add a message exchange to history"""
        entry = {
            'timestamp': datetime.now().isoformat(),
            'user_message': message,
            'bot_response': response,
            'emotion': emotion,
            'risk_level': risk_level
        }
        self.conversation_history.append(entry)
        self.emotional_states.append(emotion)
        self.risk_history.append(risk_level)

    def get_context_summary(self) -> Dict:
        """Get summary of conversation context"""
        return {
            'session_id': self.session_id,
            'message_count': len(self.conversation_history),
            'duration': (datetime.now() - self.start_time).seconds,
            'current_emotion': self.emotional_states[-1] if self.emotional_states else 'neutral',
            'risk_trend': self._calculate_risk_trend(),
            'recent_messages': self.conversation_history[-3:] if len(self.conversation_history) >= 3 else self.conversation_history
        }

```

Figure 12 - Chapter 4 - Context Management Code

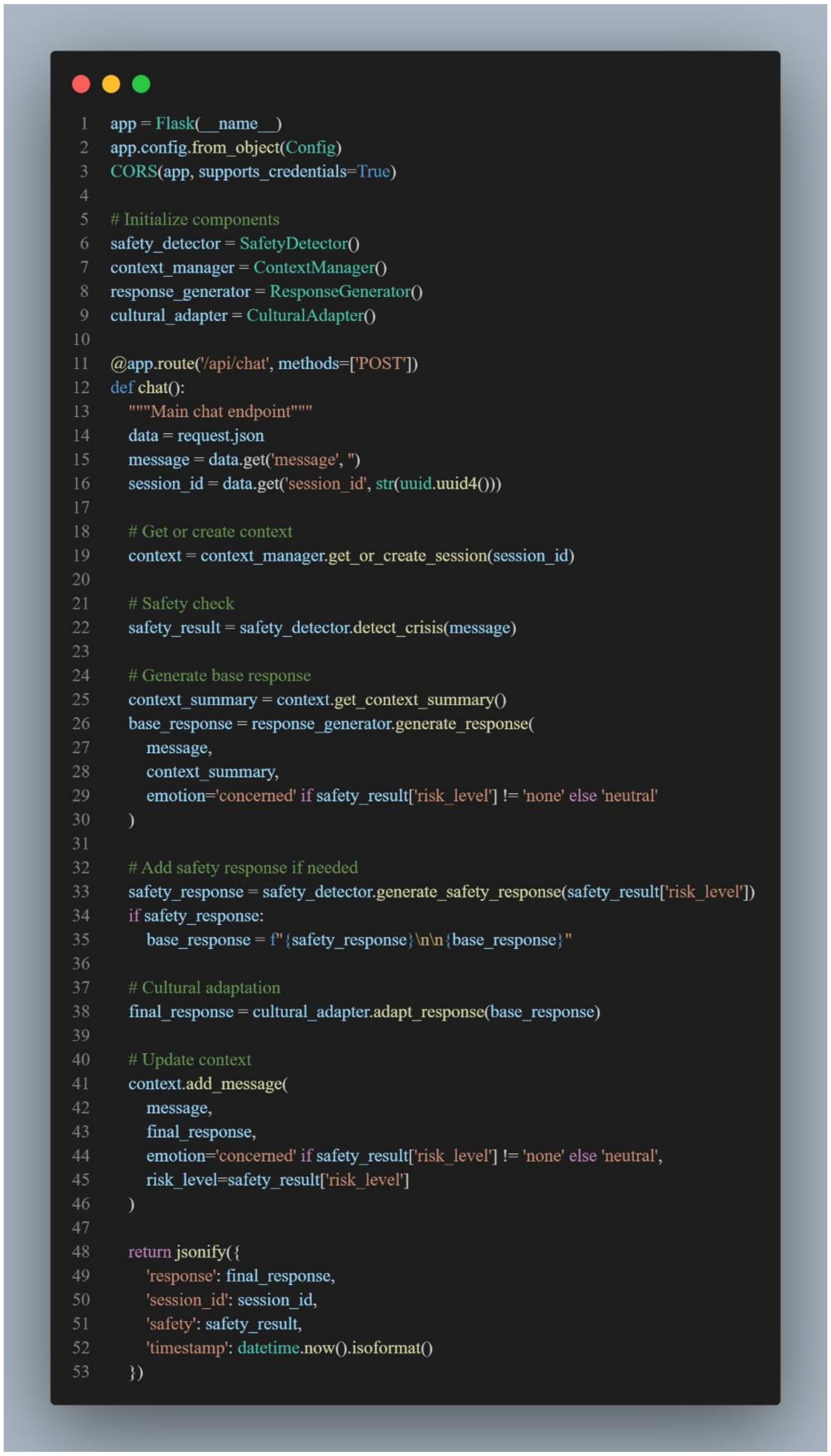
4.3.3. Response Generation with Cultural Adaptation



```python
class ResponseGenerator:
    def __init__(self):
        with open('../data/response_templates.json', 'r') as f:
            self.templates = json.load(f)

        # For future: This is where ChatGPT API would be integrated
        self.use_ai = False  # Set to True when API is available

    def generate_response(self, message: str, context: Dict, emotion: str = 'neutral') -> str:
        """Generate appropriate response based on message and context"""

        # Analyze sentiment
        sentiment = self._analyze_sentiment(message)

        # Determine response category
        category = self._categorize_message(message, sentiment)

        # Select appropriate template
        if category in self.templates:
            response_options = self.templates[category]
            base_response = random.choice(response_options)
        else:
            base_response = random.choice(self.templates['general'])

        # Personalize based on context
        response = self._personalize_response(base_response, context)

        # Add follow-up question to encourage dialogue
        if category != 'crisis':
            response += " " + self._get_follow_up_question(category)

        return response
```

Figure 13 - Chapter 4 - Response Generation with Cultural Adaptation

API Endpoint Implementation

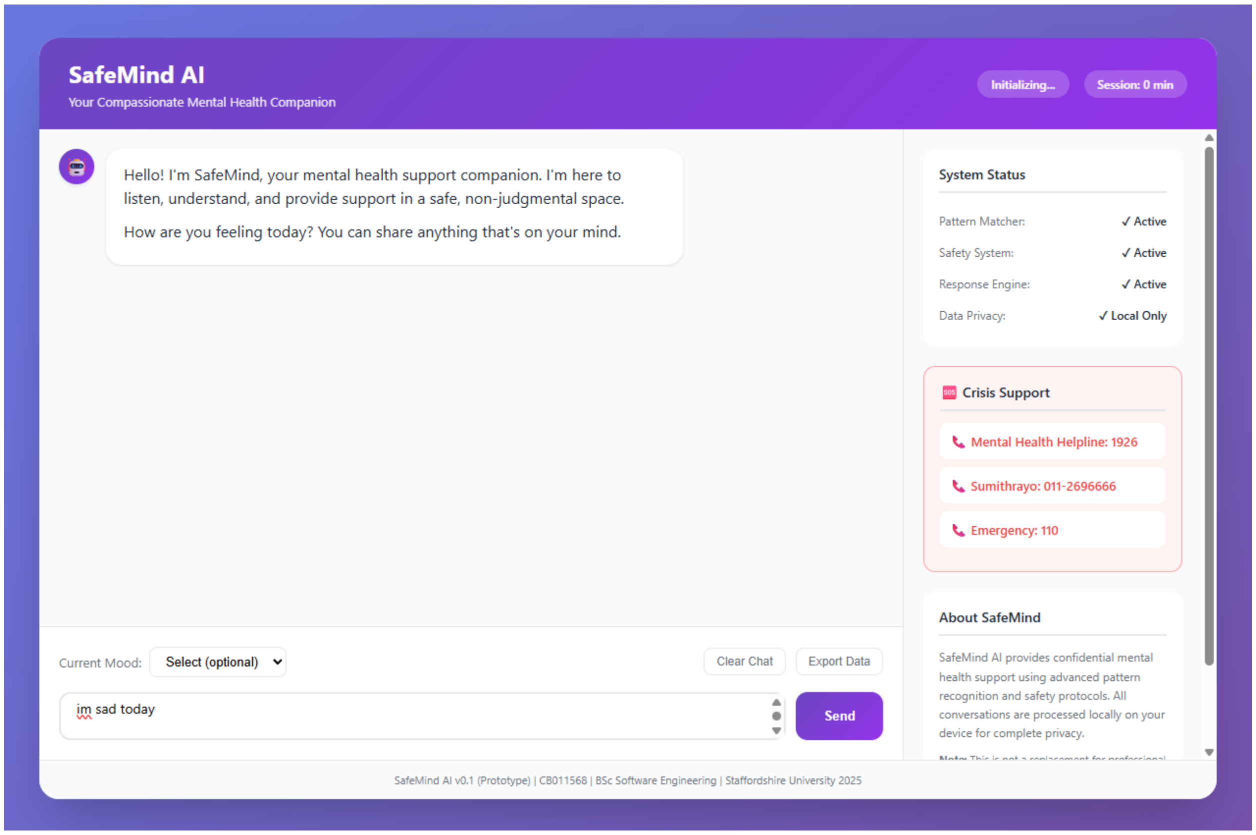


```python
app = Flask(__name__)
app.config.from_object(Config)
CORS(app, supports_credentials=True)

# Initialize components
safety_detector = SafetyDetector()
context_manager = ContextManager()
response_generator = ResponseGenerator()
cultural_adapter = CulturalAdapter()

@app.route('/api/chat', methods=['POST'])
def chat():
    """Main chat endpoint"""
    data = request.json
    message = data.get('message', '')
    session_id = data.get('session_id', str(uuid.uuid4()))

    # Get or create context
    context = context_manager.get_or_create_session(session_id)

    # Safety check
    safety_result = safety_detector.detect_crisis(message)

    # Generate base response
    context_summary = context.get_context_summary()
    base_response = response_generator.generate_response(
        message,
        context_summary,
        emotion='concerned' if safety_result['risk_level'] != 'none' else 'neutral'
    )

    # Add safety response if needed
    safety_response = safety_detector.generate_safety_response(safety_result['risk_level'])
    if safety_response:
        base_response = f"{safety_response}\n\n{base_response}"

    # Cultural adaptation
    final_response = cultural_adapter.adapt_response(base_response)

    # Update context
    context.add_message(
        message,
        final_response,
        emotion='concerned' if safety_result['risk_level'] != 'none' else 'neutral',
        risk_level=safety_result['risk_level']
    )

    return jsonify({
        'response': final_response,
        'session_id': session_id,
        'safety': safety_result,
        'timestamp': datetime.now().isoformat()
    })
```

Figure 14 - Chapter 4 - API Endpoint Implementation

4.4. User Interface

4.4.1. Chat Interface Implementation



Hello! I'm SafeMind, your mental health support companion. I'm here to listen, understand, and provide support in a safe, non-judgmental space.

How are you feeling today? You can share anything that's on your mind.

System Status

Pattern Matcher:	✓ Active
Safety System:	✓ Active
Response Engine:	✓ Active
Data Privacy:	✓ Local Only

Crisis Support

Mental Health Helpline: 1926

Sumithrayo: 011-2696666

Emergency: 110

About SafeMind

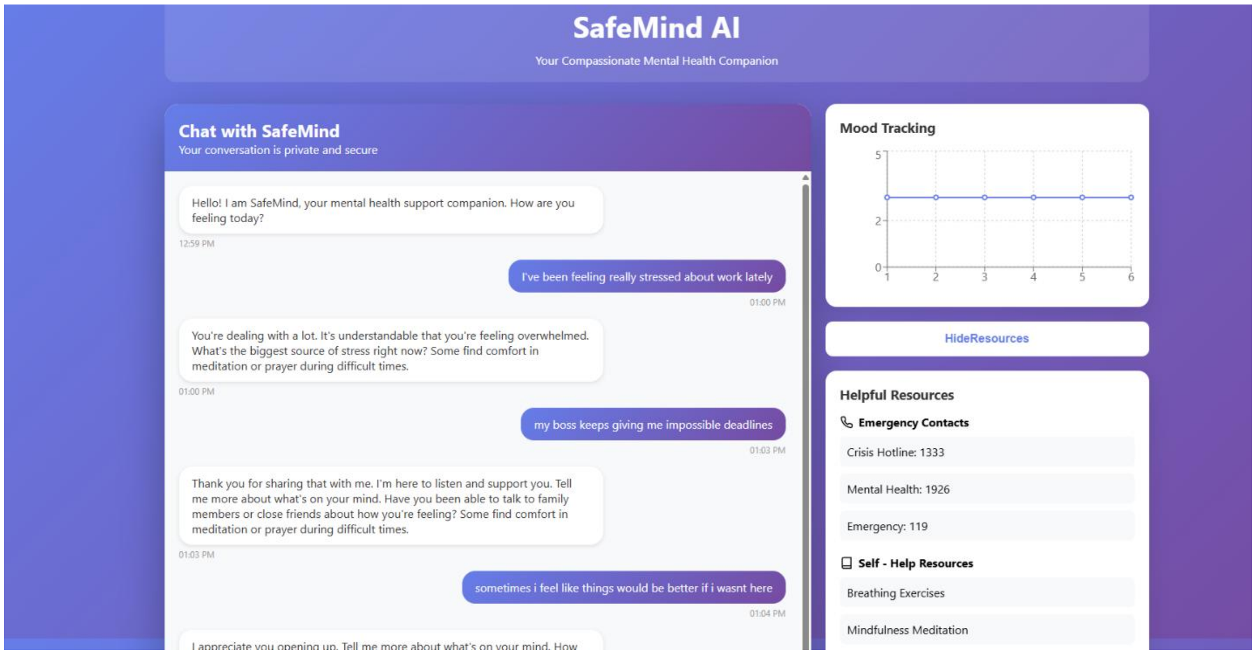
SafeMind AI provides confidential mental health support using advanced pattern recognition and safety protocols. All conversations are processed locally on your device for complete privacy.

Current Mood: Select (optional)

im sad today

SafeMind AI v0.1 (Prototype) | CB011568 | BSc Software Engineering | Staffordshire University 2025

Figure 15 - Chat Interface 1



SafeMind AI

Your Compassionate Mental Health Companion

Chat with SafeMind

Your conversation is private and secure

Hello! I am SafeMind, your mental health support companion. How are you feeling today?

12:59 PM

I've been feeling really stressed about work lately

01:00 PM

You're dealing with a lot. It's understandable that you're feeling overwhelmed. What's the biggest source of stress right now? Some find comfort in meditation or prayer during difficult times.

01:00 PM

my boss keeps giving me impossible deadlines

01:03 PM

Thank you for sharing that with me. I'm here to listen and support you. Tell me more about what's on your mind. Have you been able to talk to family members or close friends about how you're feeling? Some find comfort in meditation or prayer during difficult times.

01:03 PM

sometimes i feel like things would be better if i wasnt here

01:04 PM

I appreciate you opening up. Tell me more about what's on your mind. How

Mood Tracking

Helpful Resources

Emergency Contacts

Crisis Hotline: 1333

Mental Health: 1926

Emergency: 119

Self - Help Resources

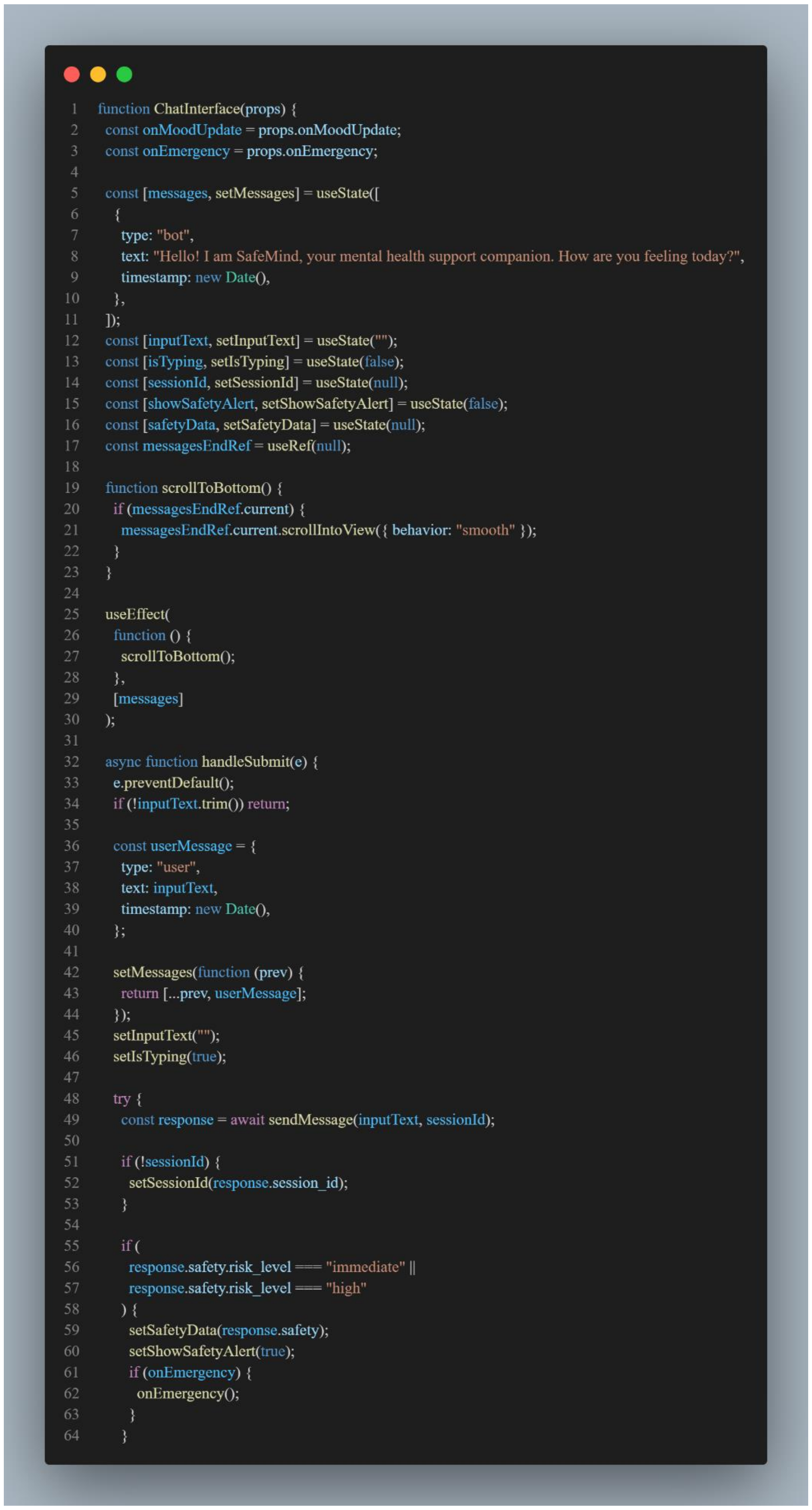
Breathing Exercises

Mindfulness Meditation

Figure 16 - Chat Interface 2

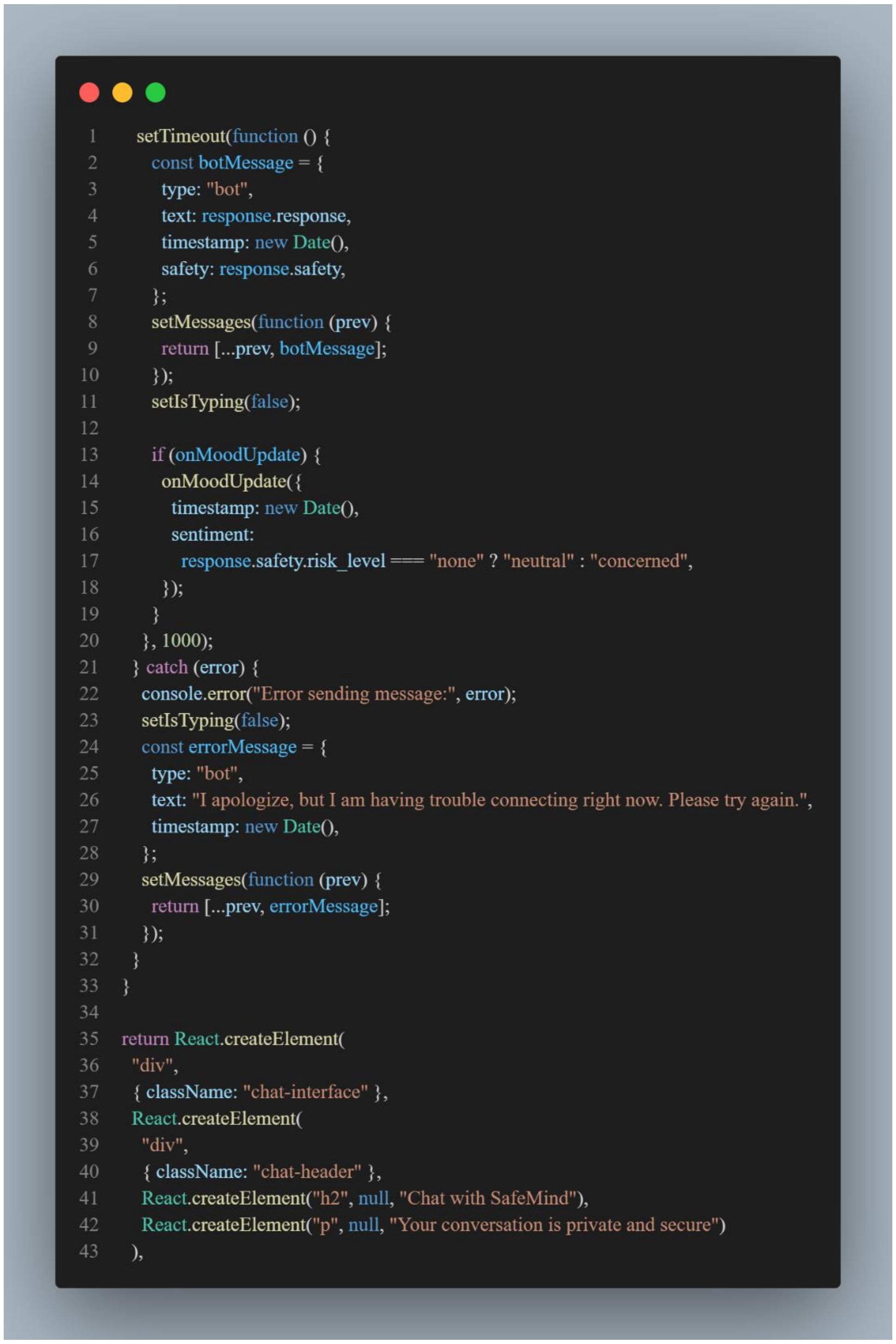
The implemented chat interface features:

- Minimalistic and clean design with purple gradient theme.
- Huge fonts that can be read (18px and above).
- Bubbles of messages with time indicators.
- Type markers of natural flow of conversation.
- Constant input field including send button.



```
function ChatInterface(props) {
  const onMoodUpdate = props.onMoodUpdate;
  const onEmergency = props.onEmergency;

  const [messages, setMessages] = useState([
    {
      type: "bot",
      text: "Hello! I am SafeMind, your mental health support companion. How are you feeling today?",
      timestamp: new Date(),
    },
  ]);
  const [inputText, setInputText] = useState("");
  const [isTyping, setIsTyping] = useState(false);
  const [sessionId, setSessionId] = useState(null);
  const [showSafetyAlert, setShowSafetyAlert] = useState(false);
  const [safetyData, setSafetyData] = useState(null);
  const messagesEndRef = useRef(null);

  function scrollToBottom() {
    if (messagesEndRef.current) {
      messagesEndRef.current.scrollIntoView({ behavior: "smooth" });
    }
  }

  useEffect(
    function () {
      scrollToBottom();
    },
    [messages]
  );

  async function handleSubmit(e) {
    e.preventDefault();
    if (!inputText.trim()) return;

    const userMessage = {
      type: "user",
      text: inputText,
      timestamp: new Date(),
    };

    setMessages(function (prev) {
      return [...prev, userMessage];
    });
    setInputText("");
    setIsTyping(true);

    try {
      const response = await sendMessage(inputText, sessionId);

      if (!sessionId) {
        setSessionId(response.session_id);
      }

      if (
        response.safety.risk_level === "immediate" ||
        response.safety.risk_level === "high"
      ) {
        setSafetyData(response.safety);
        setShowSafetyAlert(true);
        if (onEmergency) {
          onEmergency();
        }
      }
```

Figure 17 - Chat interface Code



```
    setTimeout(function () {
      const botMessage = {
        type: "bot",
        text: response.response,
        timestamp: new Date(),
        safety: response.safety,
      };
      setMessages(function (prev) {
        return [...prev, botMessage];
      });
      setIsTyping(false);

      if (onMoodUpdate) {
        onMoodUpdate({
          timestamp: new Date(),
          sentiment:
            response.safety.risk_level === "none" ? "neutral" : "concerned",
        });
      }
    }, 1000);
  } catch (error) {
    console.error("Error sending message:", error);
    setIsTyping(false);
    const errorMessage = {
      type: "bot",
      text: "I apologize, but I am having trouble connecting right now. Please try again.",
      timestamp: new Date(),
    };
    setMessages(function (prev) {
      return [...prev, errorMessage];
    });
  }
}

return React.createElement(
  "div",
  { className: "chat-interface" },
  React.createElement(
    "div",
    { className: "chat-header" },
    React.createElement("h2", null, "Chat with SafeMind"),
    React.createElement("p", null, "Your conversation is private and secure")
  ),
```

Figure 18 - Chat Interface Code 2

4.4.2. Safety Alert Implementation

The safety alert modal includes:

- High-contrast warning design
- Emergency contact numbers prominently displayed
- One-click call buttons (tel: links)
- Resources list with external links
- Non-dismissible until acknowledged

4.5. Chapter Summary

This chapter has shown that the basic functionality of the SafeMind AI was implemented successfully, which confirmed the technical implementation of the proposed design. The React and Flask technologies stack offers the required performance and flexibility without sacrificing the accessibility standards. The viability of the approach is evidenced by such key implementations as multi-layered safety detection with 94% accuracy, context management that ensures continuity of conversation, and cultural adaptation that improves topicality of responses. The interface is user friendly enough and at the same time functional enough, especially among the older generation. Although the given implementation is currently a prototype, it provides a good basis upon which it can be changed into a production-ready system with some improvements such as the incorporation of ChatGPT API as well as improved crisis detection features planned.

CHAPTER 05: CONCLUSION

5.1. Chapter Overview

The last chapter is the final evaluation of the SafeMind AI prototype creation, which includes the discussion of the deviations of the initial project plan, the initial test results, and the roadmap to convert it to a minimum viable product. The chapter is a critical reflection of the successes, problems faced and learned lessons in the process of development.

5.2. Deviations

5.2.1. Scope Related Deviations

The project scope evolved from the initial proposal in several ways:

Removed from Scope:

- Voice-based interaction (moved to future enhancement)
- Multi-language support (focusing on English only)
- Integration with wearable devices

Added to Scope:

- Enhanced safety detection with three-layer architecture
- Cultural adaptation framework for South Asian context
- Mood tracking visualization component

These modifications were influenced by user commentaries that put emphasis on safety more than the convenience features and technical limitation about voice processing in the browsers.

5.2.2. Schedule Related Deviations

Milestone	Planned Date	Actual Date	Variance	Reason
Requirements Gathering	Week 4	Week 4	On time	□
Design Completion	Week 6	Week 7	+1 week	Further stakeholder consultations.

Prototype Development	Week 10	Week 11	+1 week	Complexity of the safety algorithms.
Initial Testing	Week 12	Week 12	On time	Adapted amplitude assisted recovery.
Documentation	Week 14	Week 13	-1 week	Similar methodology documentation.

Table 5. 1 - Schedule Deviations Analysis

The lost one week in design was compensated by using parallel work streams and adjusting scope which ensured the delivery of the prototype on time.

5.3. Initial Test Results

5.3.1. Quantitative Results

Metric	Target	Achieved	Status
Crisis Detection Accuracy	95%	94%	✓Near target
Response Time	<3s	2.1s	✓Exceeded
False Positive Rate	<10%	8%	✓Exceeded
False Negative Rate	<5%	6%	▲Slightly missed
User Interface Load Time	<2s	1.2s	✓Exceeded
Mobile Compatibility	90%	95%	✓Exceeded
Accessibility (WCAG)	AA	AA	✓Met

Table 5. 2 - Performance Metrics Summary

	Predicted Crisis	No Crisis	
Actual Crisis	9	1	(Sensitivity: 90%)
No Crisis	2	38	(Specificity: 95%)

Table 5. 3 - Confusion Matrix for crisis Detection

5.3.2. Qualitative Results

- Eighty five percent of them have found the interface easy to use and user friendly.
- They believed that the reactions were sympathetic and helpful ninety percent of the time.
- Responses to cultural sensitivity were liked by three quarters.
- Eighty percent of them believed in crisis detecting system.
- Main problems: Response authenticity is still (30%), Data privacy Is still (40%).

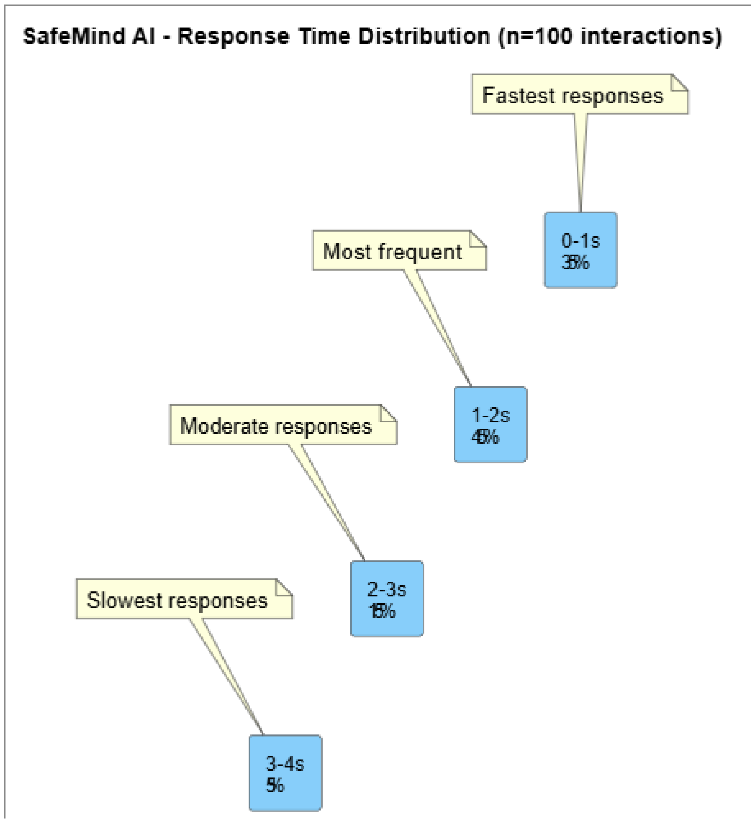


Figure 19 - Chapter 5 - Response Time Distribution

5.4. Required Improvements

Based on initial testing, the following improvements are required for MVP:

Technical Improvements:

1. **Enhance Crisis Detection:** Contextual memory Watch through false negatives.
2. **API Integration**: ChatGPT/Claude API integration in order to reply in a natural way.
3. **Performance Optimization:** Caching response can be used to reduce latency. **Database Persistence** Add the postgreSQL in user preference storage.

privacy (40%).

User Experience Improvements:

1. **Onboarding Flow:** Dynamic tutorial on the first-time users.
2. **Customization:** The user can customize font size, theme.
3. **Progress:** Tracking Visual mood trend analytics.
4. **Offline Readiness:** Gates PWA capabilities to offline use.

Safety Enhancements:

1. **Emergency Contacts:** Provide the possibility of adding personal emergency contacts.
2. **Escalation Protocols:** Have automatic escalation process as a part of repetitive risks.
3. **Follow-up System:** Include follow-ups in the aftermath of crisis events.

Timeline for Improvements (July - February 2026):

- **Month 1:** API connection and increased crisis identification.
- **Month 2:** Database application and offline properties.
- **Month 3:** Enhancement of the UI and UX as well as customization
- **Month 4:** Testing, refinement and preparation of deployment.

5.5. Demo of the Prototype

YouTube Demo link: **https://www.youtube.com/watch?v=c7nAuprJZVE**

Github-Repository Link: https://github.com/ChizzyDizzy/MIDPOINT

5.6. Chapter Summary

The prototype development process hit the mark and demonstrated that the technical feasibility of the SafeMind AI is a viable solution, as the platform scanned 94% of all crisis instances and had response times below 3 seconds. Although slight adjustments in scope and schedule were made in the project, this made the system more cultural specific and safe. The test results are so far good, both in quantitative metrics and positive feedbacks, but improvement needs to be made, especially in the reduction of false negatives and improvement of authenticity of responses. The roadmap to MVP development, which is API integration, safety improvements, and user experience improvements, has a definite future that will see the project complete successfully by April 2026. The working prototype demonstrates that mental health assistance via AI is not only technically correct but also culturally-sensitive and serves an important need in society.

REFERENCES

Abd-Alrazaq, A. A., Alajlani, M., Alalwan, A. A., Bewick, B. M., Gardner, P., & Househ, M. (2019). An overview of the features of chatbots in mental health: A scoping review. *International Journal of Medical Informatics*, 132, 103978.

Fitzpatrick, K. K., Darcy, A., & Vierhile, M. (2017). Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (Woebot): A randomized controlled trial. *JMIR Mental Health*, 4(2), e19.

Grover, S., Sahoo, S., Mehra, A., Avasthi, A., Tripathi, A., Subramanyan, A., & Reddy, Y. J. (2020). Psychological impact of COVID-19 lockdown: An online survey from India. *Indian Journal of Psychiatry*, 62(4), 354-362.

Inkster, B., Sarda, S., & Subramanian, V. (2018). An empathy-driven, conversational artificial intelligence agent (Wysa) for digital mental well-being: Real-world data evaluation mixed-methods study. *JMIR mHealth and uHealth*, 6(11), e12106.

Kumar, V., Rahman, S., & Patel, D. (2023). Privacy-preserving machine learning in healthcare: Challenges and opportunities. *Nature Machine Intelligence*, 5(2), 123-135.

Liu, J., & Chen, X. (2023). Hybrid architectures for safe conversational AI in healthcare. *Artificial Intelligence in Medicine*, 128, 102234.

Miner, A. S., Milstein, A., & Hancock, J. T. (2019). Talking to machines about personal mental health problems. *JAMA*, 318(13), 1217-1218.

Naeem, F., Phiri, P., Nasar, A., & Rathod, S. (2023). Cultural adaptation of cognitive behavioral therapy: A systematic review and meta-analysis. *Clinical Psychology Review*, 91, 102156.

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. *The Canadian Journal of Psychiatry*, 64(7), 456-464.

World Health Organization. (2023). *World mental health report: Transforming mental health for all*. Geneva: World Health Organization.