**Mental Wellness Chatbot**

**Software Requirements Specification**

**Version:** 1.0  
**Date:** September 2025  
**Prepared by:** Aarav 23106023, Aryan 23106024, Abhay 23106008, Rakshit 23106017

**Table of Contents**

1. Introduction
   * 1.1 Purpose
   * 1.2 Scope
   * 1.3 Definitions, Acronyms, and Abbreviations
   * 1.4 References
   * 1.5 Overview
2. Overall Description
   * 2.1 Product Perspective
   * 2.2 Product Functions
   * 2.3 User Characteristics
   * 2.4 Constraints
   * 2.5 Assumptions and Dependencies
3. Specific Requirements
   * 3.1 External Interface Requirements
   * 3.2 Functional Requirements
   * 3.3 Performance Requirements
   * 3.4 Design Constraints
   * 3.5 Software System Attributes
   * 3.6 Other Requirements
4. Appendices
   * A. Risk Analysis
   * B. Testing Strategy
   * C. Project Timeline

**1. Introduction**

**1.1 Purpose**

This Software Requirements Specification (SRS) defines the requirements for the Mental Wellness Chatbot, a supportive educational tool designed to provide mood check-ins, journaling prompts, coping strategies, and basic crisis detection capabilities. This document is intended for:

* Development team members
* Project supervisors and evaluators
* Future maintainers of the system

The chatbot is explicitly designed as a supplementary wellness tool and is not intended to replace professional mental health services.

**1.2 Scope**

The Mental Wellness Chatbot will be a web-based conversational system that provides:

**Primary Functions:**

* Daily mood tracking and check-ins
* Guided journaling with predefined prompts
* Coping strategy recommendations
* Crisis keyword detection with appropriate resource referrals
* Conversation history storage and export

**Benefits:**

* Accessible mental wellness support for students
* Educational tool for understanding emotional patterns
* Safe practice environment for expressing thoughts
* Crisis intervention through resource connection

**System Boundaries:** The system will operate as a standalone web application with local data storage. It will not integrate with external mental health services, medical databases, or professional counselling systems.

**1.3 Definitions, Acronyms, and Abbreviations**

**API** - Application Programming Interface  
**Chatbot** - A conversational software agent  
**Crisis Intent** - User input suggesting self-harm, suicide, or severe distress  
**Intent Classification** - Process of categorizing user input into predefined categories  
**ML** - Machine Learning  
**SPA** - Single Page Application  
**SQLite** - Lightweight relational database engine  
**SRS** - Software Requirements Specification  
**TF-IDF** - Term Frequency-Inverse Document Frequency  
**UI/UX** - User Interface/User Experience

**1.4 References**

* IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
* Kaggle Chatbot Intent Dataset for training data
* WHO Mental Health Resources documentation
* National Suicide Prevention Guidelines
* React.js Documentation (v18+)
* Flask Framework Documentation (v2.0+)
* SQLite Documentation (v3.36+)

**1.5 Overview**

This SRS is organized into three main sections following IEEE 830-1998 standards. Section 2 provides an overall description of the system including product perspective, functions, and constraints. Section 3 details specific functional and non-functional requirements. Appendices contain supporting information including risk analysis, testing strategy, and project timeline.

**2. Overall Description**

**2.1 Product Perspective**

The Mental Wellness Chatbot is a new, self-contained web application designed for educational and supportive purposes. The system architecture consists of:

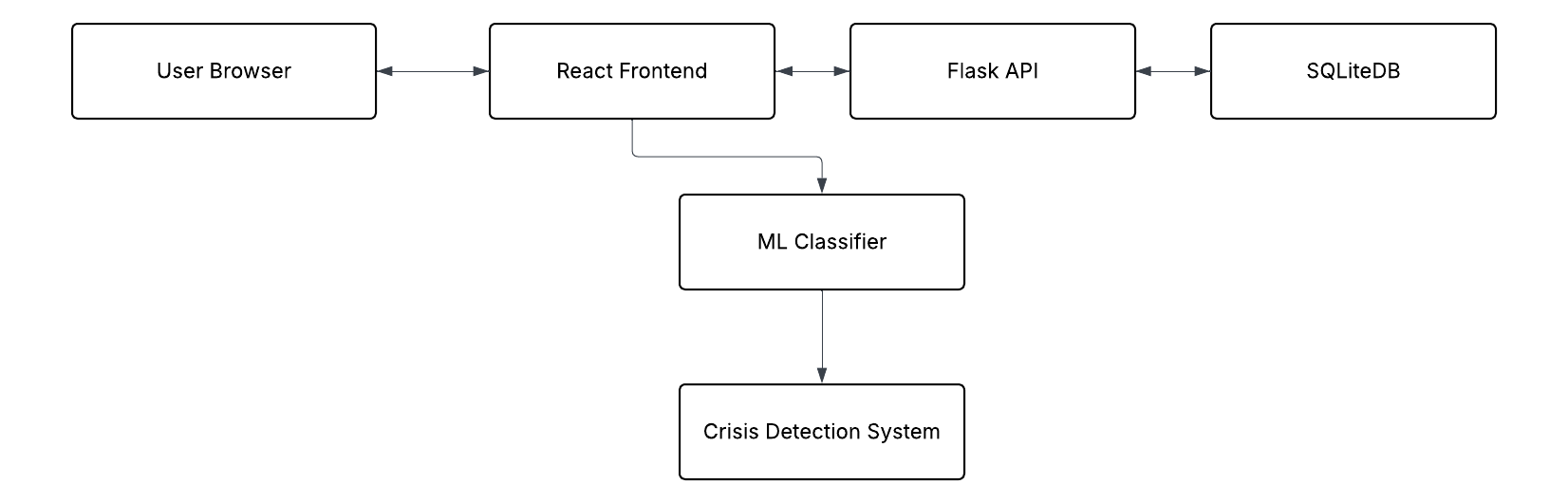
**Frontend Components:**

* React-based single-page application
* Responsive chat interface
* Data visualization for mood tracking
* Export functionality for chat history

**Backend Components:**

* Flask REST API server
* Lightweight ML classifier for intent recognition
* SQLite database for local data persistence
* Static content serving for resources

**System Architecture Diagram:**

**2.2 Product Functions**

The system provides four primary functional areas:

**F1: Mood Tracking**

* Accept user mood inputs on a 1-10 scale
* Store mood data with timestamps
* Display mood trends over time
* Provide supportive responses based on mood levels

**F2: Journaling Support**

* Offer randomized journaling prompts from predefined library
* Accept and store user journal entries
* Categorize entries by themes (gratitude, reflection, goals)
* Allow users to review previous entries

**F3: Coping Strategy Delivery**

* Provide evidence-based coping techniques
* Categorize strategies by type (breathing, mindfulness, physical activity)
* Offer personalized recommendations based on user mood patterns
* Include multimedia resources (guided audio, instructional content)

**F4: Crisis Detection and Response**

* Monitor input for crisis-related keywords and patterns
* Classify urgent situations using ML-based intent recognition
* Immediately display crisis resources and hotlines
* Escalate response appropriately without replacement of professional help

**2.3 User Characteristics**

**Primary Users: Students and Young Adults (Ages 18-25)**

* Education Level: High school to undergraduate
* Technical Expertise: Basic to intermediate computer skills
* Mental Health Awareness: Varying levels from minimal to moderate
* Usage Context: Personal wellness management, stress relief, emotional support

**User Assumptions:**

* Comfortable with web-based chat interfaces
* Seeking supplementary wellness tools, not professional therapy
* May have limited time for extended interactions
* Desire privacy and confidentiality in personal sharing

**2.4 Constraints**

**Technical Constraints:**

* Must operate on standard laptop hardware (4GB RAM minimum)
* Compatible with major web browsers (Chrome, Firefox, Safari, Edge)
* Limited to English language processing
* Local deployment required (no cloud dependencies for core functionality)

**Development Constraints:**

* 2-month development timeline
* Team of 4 undergraduate developers
* Budget constraints requiring free/open-source tools only

**Regulatory Constraints:**

* Must include appropriate disclaimers about not replacing professional help
* Crisis detection must provide immediate access to emergency resources
* Data privacy must be maintained through local storage only

**2.5 Assumptions and Dependencies**

**Assumptions:**

* Users will have reliable internet access for initial application loading
* Local browser storage will be sufficient for user data
* Predefined content libraries will meet diverse user needs
* Basic ML models will provide adequate intent classification accuracy

**Dependencies:**

* React.js framework for frontend development
* Flask framework for backend API development
* SQLite database for data persistence
* Scikit-learn library for ML model implementation
* Availability of quality training datasets for intent classification

**3. Specific Requirements**

**3.1 External Interface Requirements**

**3.1.1 User Interfaces**

**UI-1: Chat Interface** The main interface shall provide:

* Text input field with character limit of 500 characters
* Send button with keyboard shortcut (Enter key) support
* Scrollable message history with timestamps
* Clear visual distinction between user and chatbot messages
* Responsive design supporting desktop and tablet viewports

**UI-2: Dashboard Interface** The dashboard shall include:

* Mood tracking visualization with 7-day and 30-day views
* Quick access buttons for journaling and coping strategies
* Settings panel for user preferences
* Export functionality for conversation history

**UI-3: Crisis Response Interface** When crisis intent is detected, the interface shall:

* Display emergency contacts prominently
* Provide immediate access to crisis hotlines
* Show local mental health resources
* Maintain calming visual design with appropriate color schemes

**3.1.2 Hardware Interfaces**

The system shall operate on:

* Minimum 4GB RAM
* 2GB available disk space
* Network connection for initial loading
* Standard input devices (keyboard, mouse/touchpad)

**3.1.3 Software Interfaces**

**Backend Dependencies:**

* Operating System: Windows 10+, macOS 10.14+, or Linux Ubuntu 18.04+
* Python 3.8+ runtime environment
* SQLite 3.36+ database engine

**Frontend Dependencies:**

* Web Browser: Chrome 90+, Firefox 88+, Safari 14+, or Edge 90+
* JavaScript ES6+ support required

**3.1.4 Communications Interfaces**

The system shall use:

* HTTP/HTTPS protocols for client-server communication
* RESTful API design patterns
* JSON data format for API requests and responses
* WebSocket connections for real-time features (optional enhancement)

**3.2 Functional Requirements**

**3.2.1 Mood Check-in System**

**FR-1.1** The system shall accept mood ratings on a 1-10 scale with descriptive labels.

**FR-1.2** The system shall store mood data with timestamp, rating, and optional notes.

**FR-1.3** The system shall display personalized responses based on mood level:

* Ratings 1-3: Supportive messages with crisis resource checks
* Ratings 4-6: Encouraging messages with coping strategy suggestions
* Ratings 7-10: Positive reinforcement with optional gratitude prompts

**FR-1.4** The system shall generate mood trend visualizations showing patterns over time.

**3.2.2 Journaling Functionality**

**FR-2.1** The system shall provide a library of at least 50 journaling prompts categorized by:

* Daily reflection
* Gratitude practices
* Goal setting
* Emotional processing
* Creative expression

**FR-2.2** The system shall randomly select prompts or allow user selection from categories.

**FR-2.3** The system shall store journal entries with timestamps and associated prompts.

**FR-2.4** The system shall allow users to search and filter previous journal entries by date or keyword.

**3.2.3 Coping Strategy Delivery**

**FR-3.1** The system shall maintain a library of evidence-based coping strategies including:

* Breathing exercises (5 different techniques)
* Mindfulness practices (10 different activities)
* Physical movement suggestions (8 different options)
* Cognitive reframing techniques (6 different approaches)

**FR-3.2** The system shall recommend strategies based on user mood patterns and previous selections.

**FR-3.3** The system shall provide step-by-step instructions for each coping strategy.

**FR-3.4** The system shall allow users to rate strategy effectiveness for future recommendations.

**3.2.4 Crisis Detection System**

**FR-4.1** The system shall monitor all user input for crisis-related keywords including terms related to:

* Self-harm intentions
* Suicide ideation
* Severe depression indicators
* Panic or extreme anxiety

**FR-4.2** The system shall use a trained ML classifier to detect crisis intent with minimum 85% accuracy on test data.

**FR-4.3** Upon crisis detection, the system shall immediately:

* Display National Suicide Prevention Lifeline: 988
* Show Crisis Text Line: Text HOME to 741741
* Provide local emergency contact information
* Offer immediate coping resources

**FR-4.4** The system shall log crisis detection events without storing sensitive user content.

**3.2.5 Data Management**

**FR-5.1** The system shall store all user data locally using SQLite database.

**FR-5.2** The system shall provide export functionality for:

* Mood tracking data (CSV format)
* Journal entries (PDF or text format)
* Conversation history (JSON format)

**FR-5.3** The system shall allow users to delete their data completely.

**FR-5.4** The system shall implement automatic backup of database files.

**3.3 Performance Requirements**

**PR-1** The system shall respond to user input within 2-10 seconds under normal load conditions.

**PR-2** The ML classifier shall process intent classification within 500 milliseconds.

**PR-3** The system shall support concurrent usage by up to 5 users on a single deployment.

**PR-4** Database operations shall complete within 100 milliseconds for standard queries.

**3.4 Design Constraints**

**DC-1** The system must be developed using open-source technologies only due to budget constraints.

**DC-2** The ML model must be lightweight enough to run on standard laptops without GPU acceleration.

**DC-3** All user interface text must be in English only.

**DC-4** The system must operate offline after initial loading, except for crisis resource updates.

**DC-5** Database size must not exceed 100MB for reasonable performance on target hardware.

**3.5 Software System Attributes**

**3.5.1 Reliability**

The system shall maintain 95% uptime during testing periods. Error handling shall gracefully manage:

* Database connection failures
* Invalid user input
* ML model prediction errors
* Network connectivity issues

**3.5.2 Security**

The system shall implement:

* Input validation to prevent injection attacks
* Local data encryption for sensitive information
* Session timeout after 30 minutes of inactivity

**3.5.3 Maintainability**

The codebase shall:

* Follow established coding standards for Python and JavaScript
* Include comprehensive inline documentation
* Maintain separation of concerns between frontend and backend
* Provide clear error messages and logging

**3.5.4 Usability**

The interface shall:

* Require no training for basic functionality
* Provide helpful tooltips and guidance
* Support keyboard navigation for accessibility
* Maintain consistent visual design throughout

**3.5.5 Portability**

The system shall:

* Run on Windows, macOS, and Linux operating systems
* Require minimal configuration for deployment
* Use standard web technologies for maximum compatibility
* Package dependencies clearly for easy installation

**3.6 Other Requirements**

**3.6.1 Legal Requirements**

**LR-1** The system shall display appropriate disclaimers stating it is not a replacement for professional mental health care.

**LR-2** Privacy policy shall clearly explain local data storage practices.

**LR-3** Terms of use shall limit liability and set appropriate expectations.

**3.6.2 Cultural Requirements**

**CR-1** Crisis resources shall include culturally appropriate options where possible.

**CR-2** Coping strategies shall avoid cultural or religious assumptions.

**CR-3** Language used shall be inclusive and non-judgmental.

**4. Appendices**

**Appendix A: Risk Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Impact** | **Probability** | **Mitigation Strategy** |
| ML model accuracy below requirements | High | Medium | Implement keyword-based backup detection, expand training data |
| Development timeline overrun | High | Medium | Focus on MVP features, defer advanced functionality |
| Technical complexity beyond team skills | Medium | Low | Use simpler ML approaches, leverage existing libraries |
| Crisis detection false positives | Medium | Medium | Tune model thresholds, provide user feedback mechanism |
| Data loss or corruption | Medium | Low | Implement regular backups, database integrity checks |

**Appendix B: Testing Strategy**

**Unit Testing:**

* ML classifier accuracy testing with validation datasets
* Database operation testing
* API endpoint testing
* UI component testing

**Integration Testing:**

* Frontend-backend communication
* Database integration
* Crisis detection workflow
* Data export functionality

**User Acceptance Testing:**

* Usability testing with target user group
* Crisis response simulation
* Performance testing under load
* Cross-browser compatibility testing

**Appendix C: Project Timeline**

**Week 1-2: Project Setup and Research**

* Development environment setup
* Dataset collection and analysis
* UI/UX design mockups
* Technical architecture finalization

**Week 3-4: Backend Development**

* Database schema implementation
* Flask API development
* ML model training and integration
* Crisis detection system development

**Week 5-6: Frontend Development**

* React application structure
* Chat interface implementation
* Dashboard and visualization components
* Crisis response interface

**Week 7-8: Integration and Testing**

* Frontend-backend integration
* Comprehensive testing phases
* Bug fixes and optimization
* Documentation completion and deployment preparation