SereniMind - SRS

Your Personalized Path to Stress-Free Living

Software Requirements Specification (SRS)

Team - 7
Santosh Gadale
Shashank S H
Shadab Ali Khan
Shahin Nisha

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the SereniMind web application. The system is designed to help users monitor and manage their mental health by analyzing stress levels and providing personalized recommendations through interactive tools.

1.2 Scope

SereniMind is a dynamic, web-based application aimed at improving mental well-being. The system will:

- Detect stress levels using machine learning (K-Nearest Neighbors - KNN).
- Provide music therapy and exercise recommendations.
- Offer interactive quizzes and mental wellness games.
- Secure user authentication using Flask-Login.
- Use data visualization techniques for stress analysis using Plotly.
- Deploy on Heroku for cloud accessibility.

1.3 Definitions, Acronyms, and Abbreviations

- **KNN:** K-Nearest Neighbors, a machine learning algorithm for classification.
- **Flask:** A lightweight web framework for Python.
- **SQLAlchemy:** An ORM tool for database management in Python.
- **Heroku:** A cloud platform for deploying applications.
- **Plotly:** A Python library for interactive data visualization.

1.4 References

- Flask Documentation: https://flask.palletsprojects.com/
- Scikit-learn Documentation: https://scikit-learn.org/
- SQLite Documentation: https://www.sqlite.org/

1.5 Overview

This document outlines the system's core functionality, technical requirements, design constraints, and dependencies.

2. Overall Description

2.1 Product Perspective

SereniMind integrates machine learning and interactive tools to create an accessible platform for mental health monitoring and stress management.

2.2 User Classes and Characteristics

- **General Users:** Individuals seeking stress management support.
- **Administrator:** System managers responsible for authentication and monitoring.

2.3 Operating Environment

- **Backend Technologies:** Flask, SQLAlchemy, Scikit-learn.
- **Frontend Technologies:** HTML, CSS, JavaScript.
- **Database:** SQLite.
- **Hosting:** Heroku.

2.4 Design and Implementation Constraints

- Limited computational power on cloud deployment.
- Adherence to privacy and security regulations.

2.5 Assumptions and Dependencies

- Users will access the application via a web browser.
- Internet connectivity is required for full functionality.

3. Specific Requirements

3.1 Functional Requirements

- **User Authentication:** Secure login and registration using Flask-Login.
- **Stress Detection:** Machine learning-based stress assessment using KNN.
- **Visualization:** Graphical representation of stress levels using Plotly.
- **Recommendations:** Personalized suggestions for music therapy and exercises.
- **Interactive Activities:** Quizzes and mental wellness games.
- **Secure Data Management:** User data stored securely in an SQLite database.

3.2 Non-functional Requirements

- **Responsive User Interface:** Adaptive UI for desktop and mobile users.
- **Performance Efficiency:** System response time should be under two seconds for stress analysis.
- **Data Privacy:** User data should be encrypted and protected.

4. External Interface Requirements

4.1 User Interfaces

- **Dashboard:** Displays stress level analysis and recommendations.
- **Interactive Sections:** Allows users to take quizzes and play wellness games.
- **Secure Authentication:** User login and registration interfaces.
- **Graphical Representation:** Visual stress trends using Plotly.

4.2 Hardware Interfaces

- The application is designed to run on any standard device with a web browser.

4.3 Software Interfaces

- **Backend:** Flask, SQLAlchemy, Scikit-learn.
- **Frontend:** HTML, CSS, JavaScript.
- **Database:** SQLite for data storage.

4.4 Communication Interfaces

- Cloud-based deployment using Heroku.
- Internal API calls to retrieve and process stress-level data.