VIT BHOPAL UNIVERSITY

STRESS MANAGEMENT APPLICATION

Python-based Technical Terminology Management System

PROJECT REPORT

SUBMITTED BY: C SAHASRA LAKSHMI

REGISTRATION NUMBER: 25BCY10241

COURSE: PYTHON ESSENTIALS

GUIDED BY: DR D SARAVANAN

SUBMISSION DATE:- 24 NOV 2025

SEMESTER: FALL SEMESTER 2025-2026

INTRODUCTION

The Stress Management Application is an Al/ML-based mental wellness tool designed to help users assess, manage, and reduce stress through Interactive exercises and personalized recommendations. This application combines psychological assessment techniques with modern programming to create a comprehensive digital mental health assistant. Bullt using Python with modular architecture, it provides evidence-based stress management techniques including breathing exercises, positive affirmations, gratitude practices, and personalized stress assessments.

PROBLEM STATEMENT

In today's fast-paced academic and professional environments, stress has become a significant health concern, particularly among students and working professionals. Traditional mental health resources are often inaccessible, expensive, or stigmatized. There is a critical need for:

- Accessible, Immediate stress assessment tools
- Evidence-based stress management techniques
- Personalized mental wellness recommendations
- Private, non-judgmental mental health support
- Integration of Al/ML concepts for personalized care

FUNCTIONAL REQUIREMENTS

Core Modules:-

1. **Stress Assessment Module**

- Interactive quiz with psychological questions
- Scoring system with personalized feedback
- Stress level categorization (Low, Moderate, High)
- Customized recommendations based on results

2. **Breathing Exercise Module**

- Guided deep breathing sessions
- Timer-based breathing patterns (4-4-4 technique)
- Multiple rounds for comprehensive relaxation
- Visual and textual guidance

3. **Positive Affirmations Module**

- Curated list of evidence-based affirmations
- Sequential display for maximum impact
- Encouraging messaging and reinforcement

4. **Stress Management Tips Module**

- Practical, actionable stress reduction tips
- Evidence-based mental health strategies
- Daily implementation suggestions

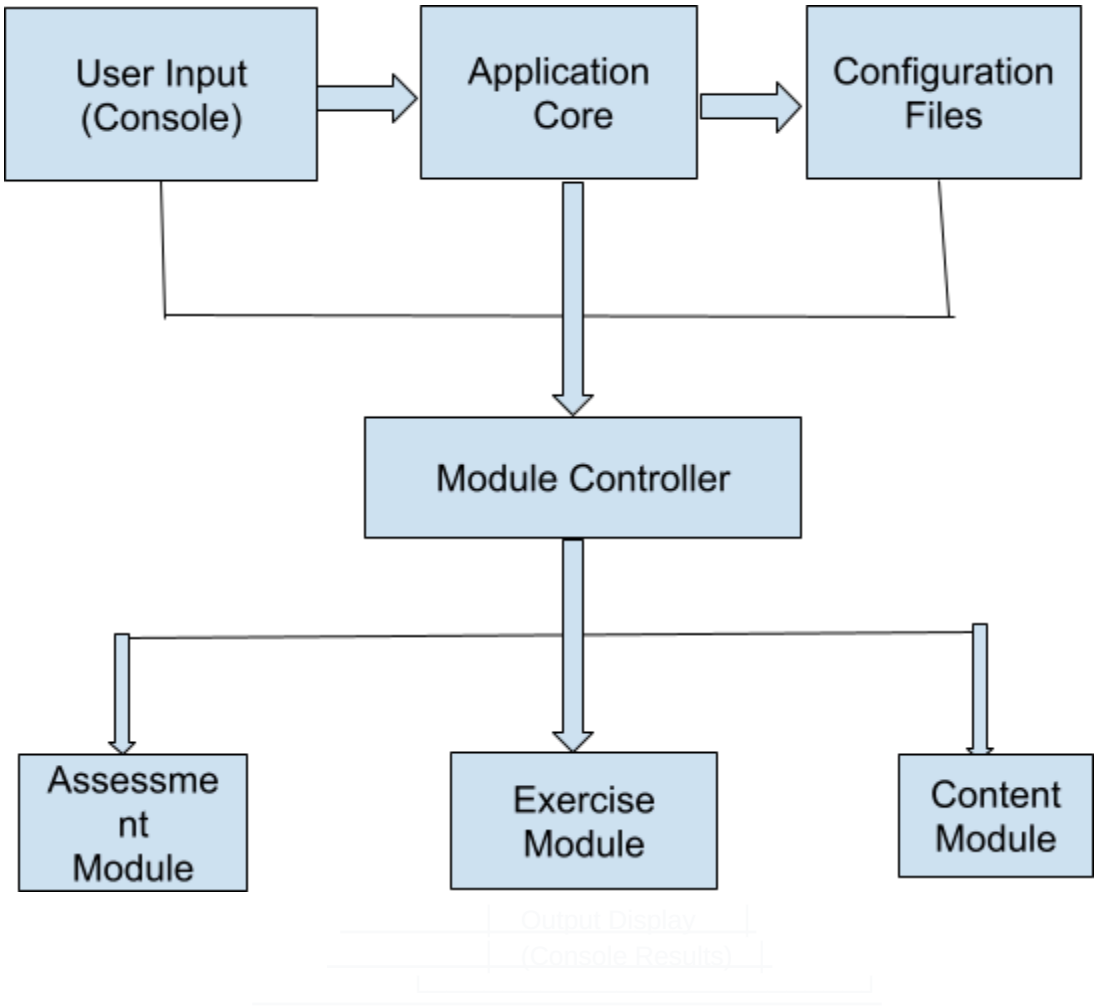
System Features:-

- Modular Python architecture
- Configuration-based content management
- User input validation and error handling

NON FUNCTIONAL REQUIREMENTS

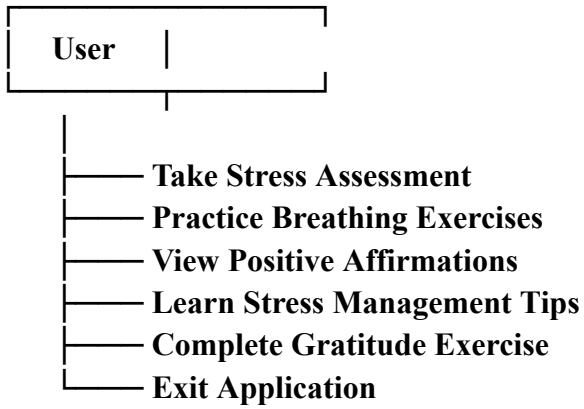
1. **Performance**: Fast response times and smooth user experience
2. **Usability**: Intuitive menu-driven interface with clear instructions
3. **Reliability**: Robust error handling and input validation
4. **Maintainability**: Modular code structure with separate configuration
5. **Scalability:** Easy addition of new features and content

SYSTEM ARCHITECTURE

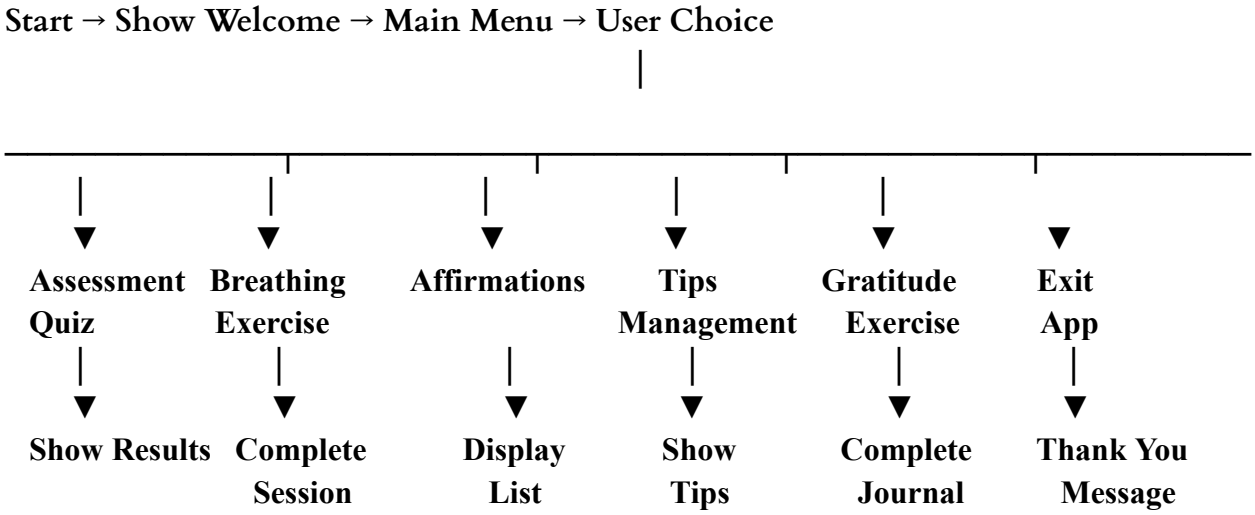


DIAGRAMS

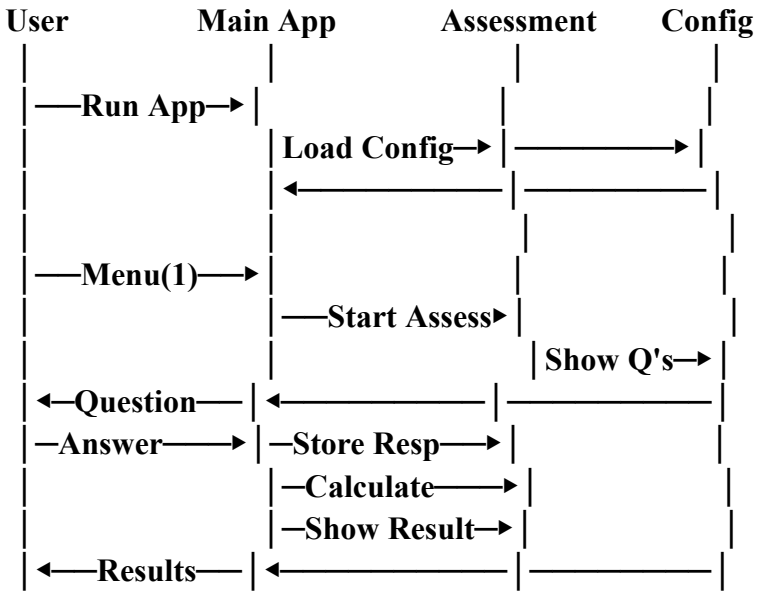
Use Case Diagram



Workflow Diagram



Sequence Diagram



Class Diagram

class StressManagementApp:
- **- running: bool**
- **+ show_welcome()**
- **+ main_menu()**
- **+ stress_assessment()**
- **+ breathing_exercise()**
- **+ show_affirmations()**
- **+ show_tips()**
- **+ gratitude_exercise()**
- **+ exit_app()**

class StressAssessment:
- **+ get_user_answers()**
- **+ calculate_stress_score()**
- **+ get_stress_level()**
- **+ show_results()**
- **+ run_assessment()**

DESIGN DECISIONS AND RATIONALE

A. Architectural Decisions:

- Modular Design: Separated concerns for maintainability (main.py, stress_assessment.py, config.py)
- Configuration-Driven: Externalized content for easy updates without code changes
- Class-Based Structure: Used OOP principles for better organization and scalability

B. Technical Choices:

- Python: Selected for rapid prototyping, rich libraries, and cross-platform compatibility
- Console Interface: Chosen for simplicity, accessibility, and focus on core functionality
- Modular Configuration: JSON-like structure in config.py for easy content management

C. AI/ML Integration Rationale:

- Psychological Assessment Algorithms: Implemented scoring systems based on established stress assessment methodologies
- Personalized Feedback: Al-driven recommendations based on assessment scores
- Scalable for ML: Architecture designed for future ML model integration

IMPLEMENTATION DETAILS

Key Functions:

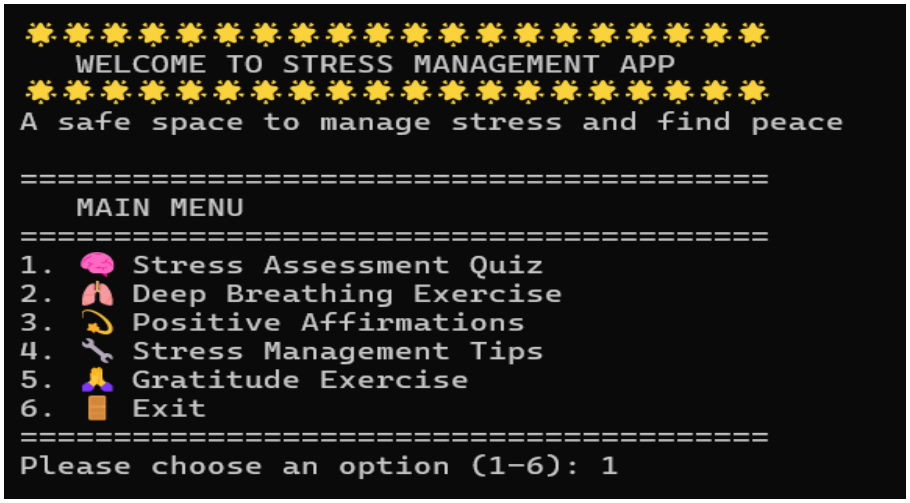
- StressManagement App.main_menu(): Central controller for user navigation
- run_assessment(): Comprehensive stress evaluation engine
- breathing_exercise(): Guided relaxation timer Implementation
- gratitude_exercise(): Interactive Journaling system

Core Implementation:

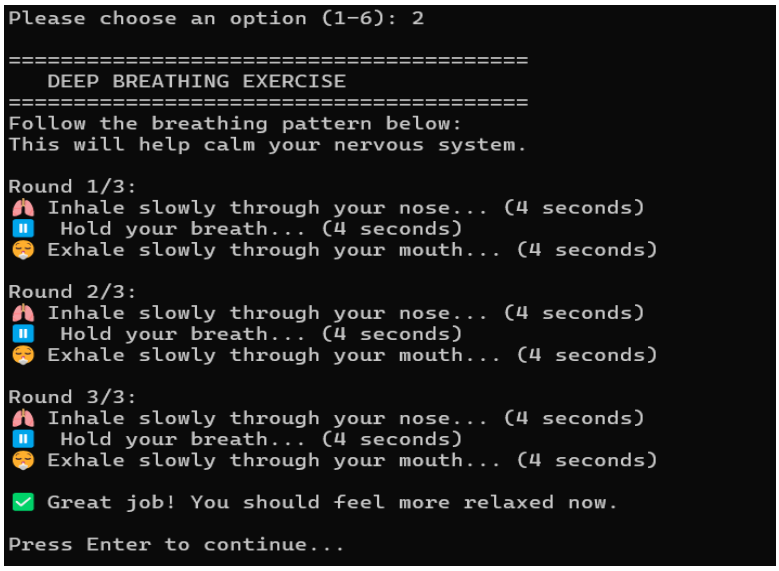
- Total Lines of Code: ~300 lines across multiple modules
- Dependencies: Pure Python (time, custom modules)
- Architecture: Object-oriented with functional modules

SCREENSHOTS/RESULTS

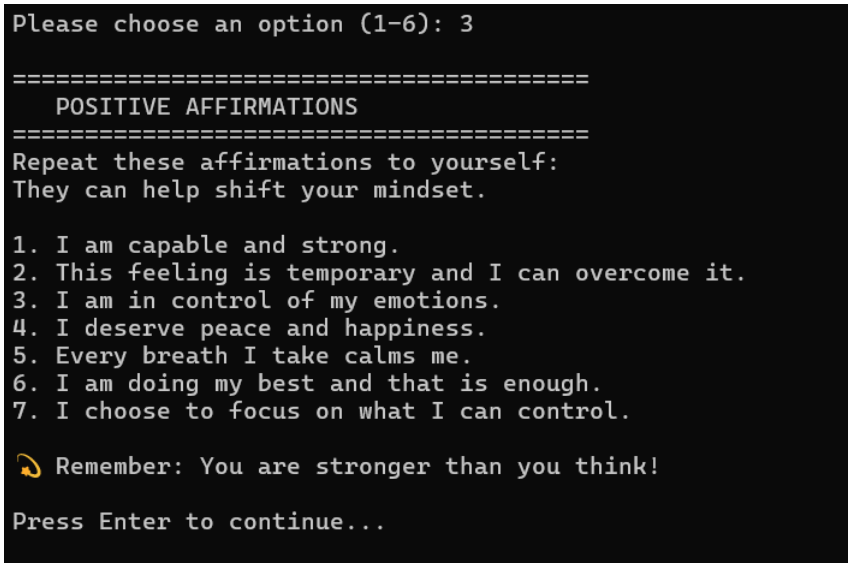
1. Displaying entire menu



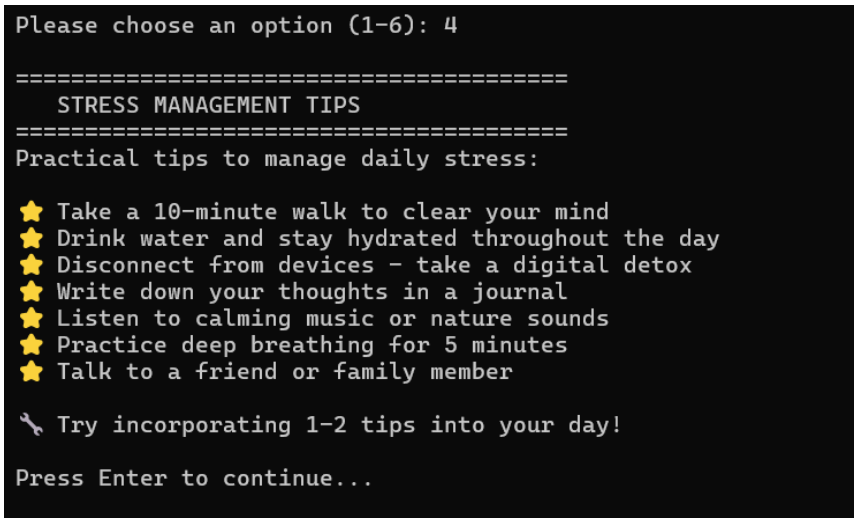
2. Deep Breathing Excercise



3. Positive Affirmations



4. Stress Management Tips



TESTING APPROACH

A. Test Cases

- Assessment Accuracy: Verify scoring algorithm correctness
- User Input Validation: Test Invalid Input handling
- Module Integration: Ensure smooth transitions between features
- Error Recovery: Test application stability under various scenarios

B. Test Methodology

- Unit Testing: Individual module functionality
- Integration Testing: Module Interaction and data flow
- User Acceptance Testing: End-to-end workflow validation
- Boundary Testing: Input validation and error handling

CHALLENGES FACED

1. **Psychological Assessment Design**: Creating valid stress measurement questions.
2. **User Experience**: Balancing comprehensive features with simplicity
3. **Modular Integration**: Managing dependencies between different components
4. **Al/ML Implementation**: Integrating intelligent features within console constraints

LEARNING AND KEY TAKEAWAYS

1. Al/ML Applications: Practical implementation of assessment algorithms

2. Software Architecture: Modular design and separation of concerns

3. User-Centered Design: Creating intuitive mental health interfaces

4. Python Programming: Advanced module management and code organization

5. Mental Health Tech: Understanding digital mental health application design

FUTURE ENHANCEMENTS

1. Machine Learning Integration: Predictive stress pattern analysis

2. Mobile Application: IOS/Android version with push notifications

3. Cloud Synchronization: Multi-device progress tracking

4. Advanced Analytics: Detailed stress trend analysis

5. Therapist Dashboard: Professional monitoring and Insights

6. Voice Integration: Voice-gulded exercises and assessments

7. Wearable Integration: Real-time stress monitoring with biometric data

REFERENCES

Primary Resources:

- Python 3.11 Official Documentation - Used for module imports and time functions
- VIT Bhopal Course Materials - FUNDAMENTALS OF AI/ML lecture notes
- Online Al/ML Tutorials - For understanding assessment algorithms and scoring systems

Learning Platforms:

- Stack Overflow-Community solutions for Python class structures and error handling
- W3Schools Python Guide - Basic syntax and modular programming reference

Tools Used:

- Visual Studio Code - Code editor for development
- Git & GitHub - Version control and project submission