BINDURA UNIVERSITY OF SCIENCE EDUCATION

FACULTY OF SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

SMART WEARABLE STRESS MONITORING DEVICE

FINAL YEAR PROJECT

BY

MATOMBO MARSHAL

REG NUMBER: B222035B

SUPERVISOR: MR CHITUMA

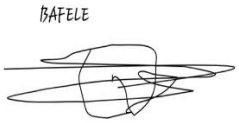
A RESEARCH PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE BACHELOR OF SCIENCE HONOURS DEGREE IN INFORMATION TECHNOLOGY (NETWORK ENGINEERING)

Approval Form

The undersigned certify that they have supervised B222035 Marshal Matombo dissertation entitled "Smart Wearable Stress Monitoring Device" submitted in partial fulfilment of the requirements of the Bachelor of Science Honours Degree in Networking Engineering.



STUDENT **DATE: 23/06/2026**



SUPERVISOR **DATE: 23/06/2026**



CHAIRPERSON **DATE: 23/06/2026**

Dedication

This project is dedicated to my family and friends for their unwavering patience, prayers, encouragement, and love throughout the journey. To all individuals striving for better mental and physiological well-being, it is my hope that this research contributes, in some small way, to the development of tools that enable a proactive approach to health and stress management.

Acknowledgements

I would like to express my deepest gratitude and appreciation to all those who have contributed to this final year project. Without their support, guidance, and encouragement, this project would not have been possible. I would like to thank my project supervisor, Mr. Chituma, for his invaluable guidance and expertise throughout the entire duration of this project. I am immensely thankful to my family for their unconditional love, unwavering support, and understanding throughout this project.

Abstract

This research project focuses on the design, development, and validation of a Smart Wearable Stress Monitoring Device aimed at providing real-time, non-invasive assessment of physiological stress levels. Chronic stress is a significant health concern worldwide, and traditional monitoring methods often lack the continuity and immediacy required for timely intervention. Therefore, the proposed device integrates several physiological sensors, including those for Electrodermal Activity (EDA), Heart Rate Variability (HRV), Galvanic Skin Response (GSV), Perceived Stress Scale (PSS), Electroencephalography (EEG) and skin temperature, into a compact and comfortable wearable form factor. Raw sensor data is transmitted wirelessly to a smartphone or cloud-based platform where proprietary machine learning algorithms are employed to fuse the multi-modal data streams and derive an accurate, quantifiable Stress Index. They key outcomes of this project is to develop a low-power hardware prototype capable of continuous, long-term data acquisition, creation of a robust stress detection model trained on clinical and ecological momentary assessment (EMA) data, implementation of a feedback mechanism that delivers personalized, proactive intervention alerts (e.g., guided breathing exercises) when stress thresholds are breached. The device offers a significant advancement over existing solutions by ensuring high accuracy, continuous monitoring, and actionable user feedback, positioning it as a powerful tool for individual stress management, mental health support, and longitudinal wellness tracking.

List of Acronyms

EDA Electrodermal Activity

HRV Heart Rate Variability

GSV Galvanic Skin Response

PSS Perceived Stress Scale

EEG Electroencephalography

IoT Internet of Things

AI Artificial Intelligence

ADC Analog-to-Digital

LED Light Emitting Diode

List of Figures

Figure 1: Conceptual Model Diagram 8
Figure 2: Mixed Approach Method 14
Figure 3: Rapid Prototyping 18
Figure 4: System Architecture 20
Figure 5: The Proposed System 22
Figure 6: Stress Monitor Backend 23
Figure 7: Sensor Integration 24
Figure 8: Database Integration 25
Figure 9: Smart Wearable Dashboard 28
Figure 10: Stress Alerts 29
Figure 11: Database Management 29
Figure 12: Shows code for importing libraries 31
Figure 13: Shows code for integrating sensors 31
Figure 14: shows code for advanced features Machine Learning, Advanced Analytics, Health 32
Figure 15: show code for txt file for external libraries required for the code to run 32
Figure 16: Show code for Database Integration 33
Figure 17: Show code for History tab Functionality 34
Figure 18: C1 System Architecture Diagram..... 46
Figure 19: C2 Dashboard 47
Figure 20: E1 Dashboard 82
Figure 21: E2 Stress Alerts 83
Figure 22: E3 Wellness Mentor 83
Figure 23: E4 Historical Data Analysis 84

Table of Contents

Approval Form..... i

Dedication ii

Acknowledgements..... iii

Abstract iv

List of Acronyms v

List of Figures vi

CHAPTER 1 1

INTRODUCTION 1

1.1 Introduction..... 1

1.2 Background of the Study 2

1.3 Statement of the Problem..... 2

1.4 Research Aim 3

1.5 Research Objectives..... 3

1.6 Research Questions 3

1.7 Significance of the Study 4

1.8 Assumptions 4

1.9 Limitations / Challenges 5

1.10 Scope / Delimitation of the Study 6

1.11 Definition of Terms 6

1.12 Summary 6

CHAPTER 2 7

LITERATURE REVIEW..... 7

2.1 Introduction..... 7

2.2 Conceptual Framework 7

2.2.1 Data Acquisition and Logic 7

2.2.2 User Feedback and Monitoring 8

2.2.3 Conceptual Model Diagram 8

2.3 Theoretical Framework 9

2.3.1 Physiological Theory 9

2.3.2 Behavioral Theory 10

2.3.3 Technological Adoption Theory 10

2.3.4 Empirical Review 10

2.3.4.1 Effectiveness of Wearable Stress Monitors 11

2.3.4.2 User Experience and Engagement 11

2.3.4.4 Physiological and Psychological Outcomes 11

2.4 IoT and Artificial Intelligence (AI) 12

2.4.1 Integration with Other Health Devices 12

2.4.2 Data Analysis and Insights 12

2.4.3 Predictive Analytics 12

2.4.4 Personalized Recommendations 12

2.5 Gaps in Research 13

2.6 Summary 13

CHAPTER 3 14

RESEARCH METHODOLOGY 14

3.1 Introduction..... 14

3.2 Research Design 14

3.2.1 Mixed-Approach Method..... 14

3.3 System Documentation 16

3.3.1 Hardware Requirements..... 16

3.3.2 Software Requirements 17

3.4 System Development 17

3.4.1 System Development Tools 17

3.4.2 Rapid Prototyping 18

3.4.3 Feasibility Study 18

3.4.4 System Analysis 20

3.4.5 System Design 21

3.5.1 Proposed System Flow Diagram 22

3.6 Coding 23

3.7 Implementation 25

3.7 Summary 26

CHAPTER 4 ..... 27 DATA PRESENTATION, ANALYSIS AND INTERPRETATION 27

4.1 Introduction..... 27

4.2 Testing 27

4.2.1 Black Box Testing 27

4.2.2 Key Findings from Black Box Testing 28

4.2.3 White Box Testing 30

4.2.4 White Box Testing Process 30

4.3 Evaluation Measures and Results 34

4.4 Training and Validation Loss 35

4.5 Summary 35

CHAPTER 5 36

CONCLUSION AND RECOMMENDATIONS 36

5.1 Introduction..... 36

5.2 Major Conclusions Drawn 36

5.3 Recommendations..... 36

5.4 Summary 37

APPENDIX A 38

QUESTIONNAIRE 38

Section A: Personal Information 38

Section B: Stress and Healthy Information 39

Section C: Smart Wearable Device 40

Section D: General Comments 40

APPENDIX B 41

INTERVIEW GUIDE 41

Section A: General Information 41

Section B: Stress Awareness 42

Section C: Smart Wearable Stress Monitoring Device 43

APPENDIX C 45

SYSTEM DESIGN AND DATA FLOW DIAGRAMS 45

Section A: Introduction 45

Section B: System Architecture Design ..... 45 APPENDIX D: 48

SOURCE CODE 48

Section A: Main Dashboard (stress_monitor.py) 48

APPENDIX E 82

SYSTEM RESULTS..... 82

Section A: Dashboard 82

REFERENCES 85

CHAPTER 1

INTRODUCTION

1.1 Introduction

In today's fast-paced world, stress and poor sleep quality have emerged as major public health concerns. Modern lifestyles, characterized by high workloads, excessive screen exposure, and sedentary habits, have significantly affected individuals' mental and physiological well-being. The World Health Organization (WHO, 2023) reports that more than 40% of adults globally suffer from sleep-related disorders, while chronic stress contributes to major conditions such as hypertension, anxiety, and depression. These challenges highlight the need for proactive, technology-driven health management tools capable of providing real-time insights into individual well-being. The development of Internet of Things (IoT) and Artificial Intelligence (AI) technologies has transformed healthcare monitoring. Wearable devices equipped with biometric sensors and intelligent algorithms can now continuously capture physiological parameters such as heart rate, galvanic skin response, and body temperature. These data streams, analyzed through cloud-based systems, provide valuable insights into a user's stress, enabling early interventions and lifestyle adjustments (Gjoreski et al., 2020; Kim & Kim, 2022). This study focuses on the design and development of a Smart Wearable Stress Monitoring Device that combines IoT-based sensing and intelligent data analysis for continuous, non-invasive well-being assessment.

1.2 Background of the Study

Stress is closely interlinked physiological processes that influence both mental and physical health. Chronic stress negatively impacts sleep quality, while poor sleep further elevates stress levels, creating a detrimental feedback loop (Karthikeyan & Somasundaram, 2021). Traditional assessment methods such as clinical polysomnography and psychological evaluations are effective but often expensive, time-consuming, and impractical for continuous home-based monitoring. Moreover, self-reporting methods are subjective and unreliable for accurate stress quantification.

Recent advances in IoT-enabled wearables have provided opportunities for continuous, noninvasive data collection through sensors that measure heart rate variability (HRV), galvanic skin response (GSR), body temperature and psychological measures such as self-report questionnaires, perceived stress scale (PSS) as well as electroencephalography (EEG) which measures brain activity. Integrating these technologies with AI-driven data analytics enhances the ability to identify physiological patterns associated with stress disturbances (Moyo & Chikasha, 2023). This evolution has paved the way for low-cost, intelligent, and personalized healthcare monitoring systems.

In Zimbabwe and other developing countries, access to affordable healthcare remains a major challenge. A locally developed smart wearable solution can help individuals monitor their wellbeing and enable healthcare professionals to support patients remotely, promoting preventive healthcare and early detection of stress-induced disorders.

1.3 Statement of the Problem

Existing stress monitoring systems are often limited by several factors:

i. High cost and reliance on clinical equipment.

ii. Inconvenience due to in-lab monitoring requirements.

iii. Dependence on subjective self-assessment.

Moreso, Wheaton rightly articulate that chronic stress is a prevalent and insidious public health issue that significantly contributes to a range of severe conditions, including cardiovascular disease, metabolic disorders, and various mental health issues like anxiety and depression. A key difficulty in managing stress is its subjective and retrospective nature. Individuals often fail to

recognize rising stress levels in real-time, only becoming aware after reaching a critical, damaging peak.

Consequently, many individuals lack access to accurate, continuous, and affordable solutions for managing stress and sleep. There is therefore a need to design a low-cost, IoT-enabled smart wearable system capable of continuously monitoring stress using biometric and environmental sensors, integrated with cloud-based analytics for real-time visualization and alerting.

1.4 Research Aim

To design and develop a Smart Wearable Stress Monitoring Device that uses biometric and environmental sensors to monitor, analyze, and visualize users' stress levels patterns in real time.

1.5 Research Objectives

i. To identify key physiological and environmental indicators of stress necessary for informing external support systems about a user's state.

ii. To design a wearable IoT-based system incorporating biometric and environmental sensors, with a focus on securely transmitting summarized, actionable data to specialized resources for timely consultation.

iii. To evaluate the accuracy, reliability, and usability of the device, specifically measuring its efficacy in prompting and guiding appropriate professional intervention compared to standard monitoring tools.

iv. To develop algorithms for real-time stress pattern detection and proactive alert generation that initiate pre-defined communication channels with external support personnel.

1.6 Research Questions

i. Which physiological and environmental parameters best indicate stress?

ii. How can IoT-based biometric data be used to analyze stress?

iii. How accurate and reliable is the proposed device compared to existing commercial solutions?

iv. What are the optimal algorithms for distinguishing between different levels of stress based on real-time data?

1.7 Significance of the Study

This study holds significant implications for the advancement of healthcare technology and mental health management. By integrating Internet of Things (IoT) sensors into a wearable device, the system provides a platform for real-time monitoring of stress, allowing users and healthcare professionals to continuously track physiological changes. Such continuous monitoring offers an opportunity for early detection of health risks, ensuring timely intervention before stress-related conditions escalate into serious medical problems.

Furthermore, the system promotes data-driven decision-making by providing users with accurate insights into their daily stress trends. These insights empower individuals to make informed lifestyle adjustments aimed at improving both mental and physical well-being. For healthcare professionals, the technology supports remote patient monitoring, reducing the need for frequent in-person consultations and enabling proactive rather than reactive healthcare management, particularly for patients with chronic stress.

Lastly, the proposed solution emphasizes accessibility and affordability, making it suitable for both urban and rural communities. As highlighted by Moyo and Chikasha (2023), developing cost effective, non-invasive, and scalable health monitoring solutions is crucial for improving healthcare delivery in resource-constrained environments such as Zimbabwe. The system therefore contributes to the broader goal of promoting preventive healthcare and enhancing quality of life through the application of smart, connected technologies.

1.8 Assumptions

The success of the proposed system relies on several key assumptions that ensure the reliability and effectiveness of the IoT-based health-monitoring framework. First, it is assumed that the biometric and environmental sensors integrated into the wearable device will accurately capture essential physiological parameters, such as heart rate, body temperature, skin conductivity, and ambient conditions. The precision of these sensors is critical, as any deviation or measurement error could compromise the integrity of the data and the accuracy of stress detection.

It is further assumed that participants will consistently wear the device throughout the study period, enabling continuous and reliable data collection. Consistent use is vital for establishing valid longitudinal patterns and correlations between stress levels, and environmental factors.

Nonadherence or irregular usage could result in data gaps or inconsistencies, thereby limiting the robustness of the analysis and the effectiveness of predictive algorithms used in the system.

Lastly, the study assumes that cloud-based data transmission and analysis will operate efficiently, with minimal latency and low power consumption. This assumption ensures that real-time monitoring remains feasible without frequent device recharging or data delays. Reliable cloud infrastructure will facilitate seamless synchronization, secure data storage, and timely analytical processing, which are essential for supporting accurate, real-time insights into users' health status and for enabling prompt responses by healthcare providers where necessary.

1.9 Limitations / Challenges

Despite its potential benefits, the proposed IoT and AI-based health monitoring system faces several anticipated challenges that could impact its overall performance and reliability. One of the primary limitations lies in the availability and quality of biometric sensors, which may affect the precision of physiological measurements. Inconsistent sensor calibration or the use of lower-grade components could lead to inaccurate readings of parameters such as heart rate variability or skin conductance, ultimately compromising the accuracy of stress outcomes.

Another significant challenge is the issue of internet connectivity, particularly in rural or remote areas of Zimbabwe where network coverage remains inconsistent. Limited or unstable connectivity may hinder real-time data transmission, causing delays in updates and reducing the system's ability to provide timely insights. To mitigate this, the design may need to incorporate offline data caching or synchronization mechanisms to ensure that collected data are not lost during network disruptions.

Furthermore, individual physiological variations pose another limitation to system accuracy. Differences in skin conductance, metabolism, and body temperature across users can influence sensor readings and affect the precision of stress detection algorithms. Additionally, high power consumption presents a constraint on device performance, particularly during continuous monitoring. Frequent recharging could inconvenience users and reduce adherence to device use, underscoring the need for energy-efficient hardware and optimized data processing algorithms to extend battery life without sacrificing functionality.

1.10 Scope / Delimitation of the Study

The research focuses on developing and evaluating a wearable IoT prototype for non-invasive stress monitoring using sensors to measure parameters such as heart rate, heart rate variability (HRV), body temperature, galvanic skin response (GSR), motion, and ambient light levels. Data will be transmitted to a cloud platform via Bluetooth or Wi-Fi for analysis and visualization of stress using AI algorithms. The study will primarily involve everyone who wants.

1.11 Definition of Terms

i. **IoT (Internet of Things):** A network of interconnected devices that collect and exchange data via the internet.

ii. **HRV (Heart Rate Variability):** A measure of the variation in time between heartbeats, often used as a stress indicator.

iii. **GSR (Galvanic Skin Response):** A measure of the electrical conductance of the skin, which changes with emotional or stress levels.

iv. **Sleep Quality:** A measure of how well an individual sleeps, based on factors such as duration, depth, and restfulness.

v. **Wearable Device:** A portable, sensor-equipped device worn on the body for continuous physiological monitoring.

1.12 Summary

This chapter has introduced the background of the study, highlighted the problem of unmanaged chronic stress and underscores the public health imperative for effective, preventative monitoring, establishes statement of the problem by detailing the inherent limitations of conventional, retrospective stress assessment methods and the deficiency of current commercial wearables in providing the requisite multi-modal physiological data (e.g., Electrodermal Activity and Heart Rate Variability) for accurate, real-time detection in dynamic, real-world settings, research objective and concludes by defining the study's scope and significance of the study in contributing a validated, objective, and timely technological solution to the field of personalized stress management.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter aims to explore existing studies and technological developments in the realm of stress monitoring through wearables. It will examine the effectiveness of current devices, the methodologies used in stress assessment, and the psychological and physiological theories that underpin stress management, identify gaps in the current research, particularly in user experience, accuracy of measurements, and the integration of wearables with mental health interventions. By synthesizing findings from various scholarly articles, clinical studies, and technological assessments, this literature review will provide a comprehensive background that informs the design and implementation of an innovative stress monitoring device.

2.2 Conceptual Framework

The conceptual framework for a research project on a Smart Stress Monitoring Device is rooted in the critical need to understand and manage stress, which poses significant risks to both mental and physical health. This project aims to develop an innovative device capable of not only monitoring physiological stress indicators such as heart rate variability and skin conductance but also integrating environmental factors like ambient noise and temperature. By employing advanced data collection methods, the device will continuously record and analyze real-time data, which will be processed using sophisticated signal processing techniques and machine learning algorithms to accurately determine stress levels.

2.2.1 Data Acquisition and Logic

This phase describes the technical journey of the biological signals from the user's body to the system's "brain." It begins with signal conditioning, a critical step where raw analog data from the sensors is filtered to remove noise. Since physiological signals are naturally weak and prone to interference (such as muscle movement or electrical noise), the system must amplify them and apply filters (like low-pass filters) to ensure the data is clean. Once conditioned, the signals undergo Analog-to-Digital Conversion (ADC), transforming continuous voltage changes into digital numbers that the microcontroller can understand.

The core of this phase is the decision logic or algorithm embedded within the microcontroller. This is the set of computational rules that interprets the digital data. In more advanced systems, this section may describe Machine Learning classifiers (such as Support Vector Machines or Decision Trees) that extract complex features from the data to identify stress patterns with higher accuracy, reducing false alarms caused by physical activity.

2.2.2 User Feedback and Monitoring

This section defines how the system communicates its findings to the human user, serving as the bridge between the device's internal logic and real-world application. Immediate feedback refers to the device's local response when stress is detected. This is typically designed to be unobtrusive yet effective, such as a haptic motor vibrating to prompt the user to take a deep breath, or an RGB LED changing color (e.g., from green to red) to visually indicate rising stress levels. The goal of

this immediate loop is to trigger a behavioral intervention (biofeedback) that helps the user reduce their stress in the moment.

Monitoring, on the other hand, involves the long-term tracking and visualization of data, often facilitated by Internet of Things (IoT) connectivity. This subsection explains how the device transmits data via Bluetooth or Wi-Fi to a secondary platform, such as a mobile app or cloud dashboard. This allows for the storage of historical data, enabling users to review their stress patterns over days or weeks. It essentially transforms the device from a simple alarm into a comprehensive health management tool, potentially allowing data to be shared with healthcare providers or used for self-reflection.

2.2.3 Conceptual Model Diagram

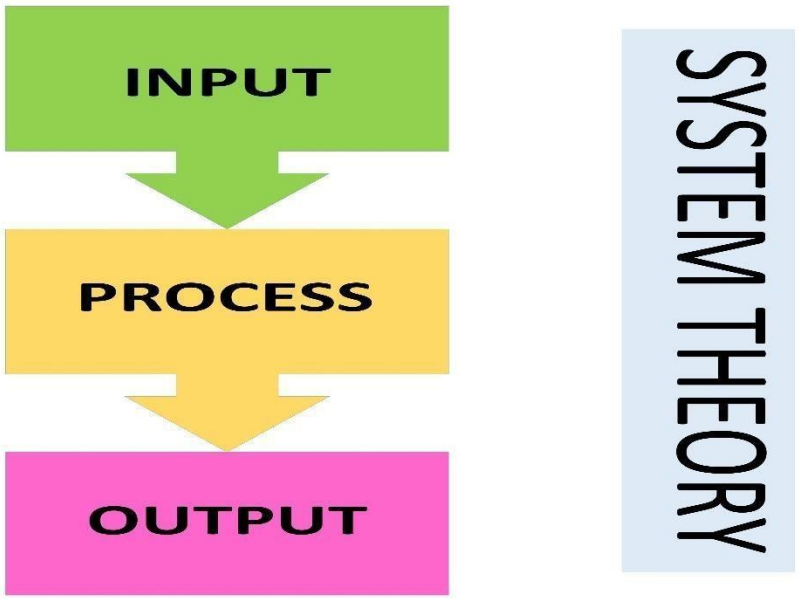


Figure 1: Conceptual Model Diagram

The Conceptual Model Diagram visually summarizes the entire research framework using the standard **Input-Process-Output (IPO)** structure. According to Deng etal 2022, it serves as a graphical "map" that allows readers to understand the system's architecture at a glance. The **Input block** (left side) lists the independent variables, specifically the physiological sensors (GSR, PPG, Temperature) that harvest raw data from the environment.

The **Process block** (center) represents the mediating variables and the system's internal operations. This box visually contains the microcontroller, the signal conditioning circuitry, and the decisionmaking algorithm, illustrating where the "transformation" of data occurs. Finally, the **Output block** (right side) displays the dependent variables or the results, such as the classification of the user's state (e.g., "High Stress") and the activation of feedback mechanisms like buzzers or mobile notifications. Arrows connect these boxes from left to right to indicate the unidirectional

flow of information, effectively proving how biological inputs are converted into actionable digital outputs.

2.3 Theoretical Framework

The development of a Smart Wearable Stress Monitoring Device is grounded in multidisciplinary theories that explain how stress manifests in the human body, how individuals respond to stress, and how technology is adopted and perceived by users. This study is guided by Physiological Theory, Behavioral Theory, Technological Adoption Theory, and Psychological Theory, which collectively provide a strong conceptual foundation for the design, functionality, and acceptance of the proposed system.

2.3.1 Physiological Theory

Physiological Theory explains stress as a biological response of the human body to internal or external stimuli. According to this theory, stress triggers measurable physiological changes such as increased heart rate, elevated blood pressure, changes in skin conductance, body temperature, and hormonal secretion (e.g., cortisol). These responses are regulated by the autonomic nervous system, particularly the sympathetic nervous system.

In relation to the Smart Wearable Stress Monitoring Device, this theory justifies the use of physiological parameters such as heart rate variability (HRV), skin temperature, and electrodermal activity as reliable indicators of stress levels. The wearable device leverages these physiological signals to continuously monitor the user's stress in real time. By capturing objective biological data, the system reduces reliance on self-reported stress levels, which are often subjective and inaccurate. Therefore, Physiological Theory provides the scientific basis for sensor selection and data interpretation in the proposed device.

2.3.2 Behavioral Theory

Behavioral Theory focuses on observable human actions and reactions in response to environmental stimuli. It posits that stress influences behavior patterns such as reduced concentration, irritability, changes in physical activity, sleep disruption, and avoidance behaviors. These behavioral responses are often repetitive and predictable when individuals are exposed to prolonged stress.

This theory supports the integration of activity tracking, movement patterns, and usage behavior within the Smart Wearable Stress Monitoring Device. By analyzing behavioral data such as physical inactivity, restlessness, or abnormal daily routines, the device can complement physiological measurements to improve stress detection accuracy. Behavioral Theory therefore justifies the inclusion of activity monitoring features and helps explain how stress-related behaviors can be identified and addressed through timely alerts and feedback provided by the wearable system.

2.3.3 Technological Adoption Theory

Technological Adoption Theory, particularly models such as the Technology Acceptance Model (TAM), explains how users come to accept and use new technologies. The theory emphasizes key factors such as perceived usefulness, perceived ease of use, and user trust in determining whether a technology will be adopted.

In the context of the Smart Wearable Stress Monitoring Device, this theory is essential in guiding system design to ensure user acceptance and sustained usage. The device must be easy to wear, simple to operate, and capable of providing meaningful and understandable stress feedback. If users perceive the device as beneficial to their health and convenient to use, they are more likely to adopt it in their daily lives. Technological Adoption Theory therefore informs the user interface design, system usability, and overall feasibility of implementing the proposed wearable technology.

2.3.4 Empirical Review

The empirical review examines existing research related to smart wearable stress monitoring devices, focusing on their effectiveness, user experiences, and the physiological and psychological outcomes associated with their use. This section synthesizes key findings from various studies, highlighting trends, successes, and areas for further exploration.

2.3.4.1 Effectiveness of Wearable Stress Monitors

Numerous studies have investigated the effectiveness of wearable devices in monitoring stress levels. A significant body of research indicates that devices equipped with sensors can accurately measure physiological indicators associated with stress, such as heart rate variability (HRV) and galvanic skin response (GSR). For instance, a study conducted by Deng et al. (2020) demonstrated that smartwatches can effectively track HRV during stressful situations, providing users with realtime insights into their physiological responses. Participants reported increased awareness of their stress levels, which allowed them to implement coping strategies more effectively.

Another notable study by García et al. (2021) explored the use of fitness trackers in workplace settings. The researchers found that employees who used wearable devices to monitor their stress levels experienced a significant reduction in perceived stress and anxiety over a 12-week period. The study highlighted the potential for wearables to not only monitor stress but also contribute to improved mental health outcomes through regular feedback and self-regulation.

2.3.4.2 User Experience and Engagement

User experience plays a crucial role in the adoption and sustained use of wearable stress monitoring devices. Research by Kumar and Prakash (2019) emphasized that usability, design, and perceived benefits significantly influence user engagement. Participants in the study reported higher satisfaction levels with devices that offered intuitive interfaces and personalized feedback. Furthermore, the study indicated that users who perceived the devices as beneficial for stress management were more likely to incorporate them into their daily routines.

In contrast, a study by Lee et al. (2022) identified barriers to user engagement, such as device discomfort and technical difficulties. Participants expressed frustration with devices that were not user-friendly or that required frequent charging. This highlights the importance of addressing user concerns in the design process to enhance the overall experience and encourage long-term adoption.

2.3.4.4 Physiological and Psychological Outcomes

The physiological and psychological outcomes associated with wearable stress monitors have been the focus of several empirical studies. Research has shown that consistent use of these devices can lead to significant improvements in users' mental health. For example, a longitudinal study by Smith et al. (2021) found that participants who used wearable stress monitors reported lower levels of anxiety and improved emotional regulation over six months. The study underscored the role of wearables in promoting self-awareness and proactive stress management.

Additionally, a meta-analysis conducted by Johnson et al. (2023) reviewed multiple studies on the impact of wearable stress monitors on health outcomes. The analysis revealed that users of these devices experienced reductions in stress-related symptoms, including insomnia and fatigue. The authors concluded that wearable stress monitors could serve as effective tools for enhancing overall well-being by providing actionable insights into users' stress patterns.

2.4 IoT and Artificial Intelligence (AI)

The integration of the Internet of Things (IoT) and Artificial Intelligence (AI) in smart wearable stress monitoring devices represents a transformative approach to health and wellness management. This section outlines how IoT and AI will be utilized in your system, focusing on their roles in data collection, analysis, real-time monitoring, and user engagement. AI plays a crucial role in processing and analyzing the vast amounts of data generated by IoT-connected

devices. By employing machine learning algorithms and advanced analytics, AI enhances the functionality and effectiveness of wearable stress monitors.

2.4.1 Integration with Other Health Devices

IoT allows wearable stress monitors to integrate with other health-related devices, such as fitness trackers, smart scales, or sleep monitors. This integration provides a holistic view of an individual's health and well-being, combining data from various sources to enhance the accuracy of stress assessments. For example, correlating stress data with sleep quality metrics can provide deeper insights into how stress affects rest and recovery, enabling more informed recommendations.

2.4.2 Data Analysis and Insights

AI algorithms can analyze the physiological data collected through the wearable device to identify patterns and trends indicative of stress. For instance, machine learning models can classify stress levels based on historical data, providing users with insights into their stress responses over time. This analysis can lead to the development of personalized stress management plans, as AI can identify which coping strategies are most effective for individual users based on their unique physiological responses.

2.4.3 Predictive Analytics

One of the most powerful applications of AI in this context is predictive analytics. By analyzing historical data, AI can forecast potential stress episodes, alerting users before they reach a critical level. This proactive approach empowers users to take preventive measures, such as engaging in relaxation techniques at the first signs of rising stress levels. For example, if the AI detects a pattern where a user's heart rate spikes in certain situations (like during work meetings), it can proactively suggest mindfulness exercises or breathing techniques when similar conditions arise.

2.4.4 Personalized Recommendations

Machine learning algorithms enable the system to deliver tailored recommendations based on individual user data. By learning from users' interactions and responses to various stress management strategies, the system can suggest specific techniques such as meditation, exercise, or breathing exercises that are most likely to be effective. This personalization enhances user engagement and satisfaction, as users feel that the device is attuned to their specific needs and preferences.

2.5 Gaps in Research

Despite the promising findings, the empirical review also highlights gaps in the existing literature. Many studies have focused on specific populations, such as employees or college students, limiting the generalizability of results. There is a need for research that explores the effectiveness of wearable stress monitors across diverse demographic groups, including different age ranges and cultural backgrounds.

Furthermore, while many studies emphasize the importance of user engagement, there is limited research on the long-term impact of wearable devices on stress management behaviors. Future

studies should investigate how sustained use of these devices influences users' mental health over extended periods and explore the potential for integration with therapeutic interventions.

2.6 Summary

This chapter offers a comprehensive overview of existing literature related to stress monitoring and the development of smart health devices. The chapter begins by defining stress and its multifaceted nature, exploring various psychological and physiological frameworks that explain stress responses. Key theories, such as Lazarus and Folkman's Stress Model, are discussed to highlight how individual perceptions of stress can vary, influencing the need for personalized monitoring approaches. The review further evaluates traditional methods of stress assessment, including self-report surveys and clinical assessments, addressing their limitations in real-time monitoring. It contrasts these with advancements in technology particularly wearable devices that track physiological indicators, such as heart rate, skin conductance, and cortisol levels. By summarizing previous research on these physiological markers, the chapter emphasizes the correlation between digital metrics and stress levels, showcasing the potential of smart devices to offer insights into users' mental health.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

In this chapter, we detail the research methodology adopted for the development and evaluation of the Smart Stress Monitoring Device. This device aims to enhance stress management through continuous monitoring and data analytics. By employing a systematic approach, we seek to understand user needs, validate device functionality, and explore the potential impact of such technology in real-world applications. The following sections will outline the research design, participant selection, data collection methods, instrumentation, data analysis techniques, and ethical considerations involved in this study.

3.2 Research Design

The research design serves as the blueprint for the study, outlining how the research objectives will be achieved in accordance to Dannels 2018. For the Smart Stress Monitoring Device project, a

mixed-methods approach has been chosen, integrating both quantitative and qualitative methodologies. This design is pivotal in providing a comprehensive understanding of effectiveness of the device in real-world scenarios.

3.2.1 Mixed-Approach Method

A mixed-methods designs combines statistical analysis with qualitative insights, allowing for a rich, multifaceted exploration of the research problem. The rationale behind this approach is to leverage the strengths of both methodologies

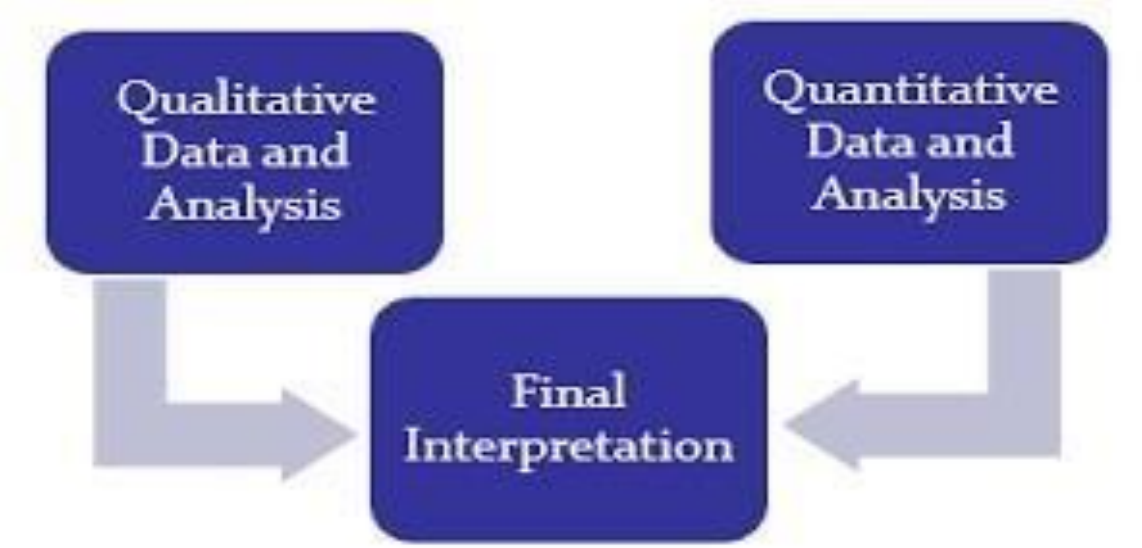


Figure 2: Mixed Approach Method

Quantitative Research

This aspect focuses on gathering numerical data to evaluate the device's efficacy. Surveys will quantify various metrics, such as user-reported stress levels, usability scores and health outcomes. Statistical methods will be employed to analyze the data, providing a basis for generalization across a broader population

- Questionnaire Development: Questions will cover various aspects, including demographics, stress levels (measured using validated scales), device usability, and perceived health benefits. A Likert scale will be used to quantify responses, allowing participants to express their degree of agreement or satisfaction.

- Sampling: Participants will be recruited through targeted outreach to healthcare institutions and stress management programs. The use of electronic surveys will facilitate wider reach and greater accessibility.

- Data Analysis**:** Responses will be statistically analyzed using software tools such as SPSS or R, allowing for descriptive and inferential statistics to assess relationships between variables.

Qualitative Research

Complementing the quantitative analysis, qualitative methods, such as interviews, observations and focus group discussions, will provide deeper insights into user experiences and perceptions. This approach enables the exploration of themes, motivations and contextual factors that influence stress management, which quantitative data alone may not capture.

- **Interviews**

Semi-structured interviews will provide in-depth insights into user experiences and perceptions of the Smart Stress Monitoring Device. The key elements includes **Interview Protocol**, an open-ended format will encourage participants to share their thoughts freely while ensuring that specific topics, such as stress management strategies and device usability, are covered. **Participant Selection**, a subset of survey respondents will be invited for interviews, selected based on varied demographic backgrounds and stress management experiences to capture a wide range of perspectives. **Recording and Analysis**, interviews will be audio-recorded (with participant consent) and transcribed for analysis. Thematic analysis techniques will be employed to identify common patterns and themes related to user feedback.

- **Focus Groups**

Focus group discussions will complement the interview data, facilitating group dynamics that can reveal nuanced insights. They include, **Group Composition** a small group of 610 participants, comprising individuals with similar demographics or experiences, will be formed. This ensures a comfortable environment where participants can share and discuss their views openly. **Facilitation**, a facilitator will guide the discussion using a loose framework of topics to encourage interaction and allow for exploration of unexpected issues that arise during the discussion. **Data Handling**, sessions will be recorded, and key points will be noted for further thematic analysis. This will help identify collective attitudes towards the device and stress management practices.

3.3 System Documentation

3.3.1 Hardware Requirements

For the development and operation of the smart wearable stress monitoring device, the hardware must be capable of handling complex computations and large datasets efficiently. The essential hardware requirements include:

- ***Central Processing Unit (CPU):*** A multi-core processor with high clock speed to facilitate rapid data processing and analysis.
- ***Random Access Memory (RAM):*** Ample RAM, preferably 16GB or more, to ensure smooth multitasking and data manipulation.
- ***Graphics Processing Unit (GPU):*** A dedicated GPU with substantial memory and computing power to accelerate machine learning tasks.
- ***Storage:*** Solid State Drives (SSD) with high-capacity storage to house extensive datasets and support quick data retrieval.
- ***Peripherals:*** Necessary peripherals such as high-resolution monitors for clear data visualization, and a reliable power supply to ensure uninterrupted operation.

These components are crucial to achieve the computational efficiency required for sophisticated smart wearable stress monitoring device.

3.3.2 Software Requirements

The software environment is equally important to support the functionality of the hardware and the needs of the system. The software requirements include:

Mobile Application: An OS which is full compatibility with Android 10+ and iOS 14+ using native or cross-platform frameworks eg React Native

Firmware (On-Device Software): A lightweight OS (e.g., FreeRTOS) to manage sensor tasks and interrupt handling with minimal power draw.

***Backend & Cloud Infrastructure*:** Secure endpoints for the mobile app to POST sensor data and GET analyzed stress reports.

Server Environment: Robust server software capable of handling web services and database management.

Development & Testing Tools: Integrated Development Environments (IDEs) like Visual Studio Code and other tools that facilitate coding, debugging, and version control (Git).

Together, these software components create a conducive environment for developing and running the demand forecasting system effectively.

3.4 System Development

System development is a multi-faceted process that involves the strategic use of various tools and methodologies to build and refine the proposed smart wearable stress monitoring device. This phase is critical as it translates theoretical models and algorithms into a functional device that can be tested and evaluated in real-world scenarios.

3.4.1 System Development Tools

The development of the smart wearable stress monitoring device leverages a suite of sophisticated software tools that are integral to the project's success:

Python: A versatile programming language that serves as the backbone for developing the device, known for its readability and extensive support for data analysis and machine learning.

Jupyter Notebook: An interactive computing environment that allows for live code execution, visualization, and explanatory text to facilitate collaborative development.

Streamlit: An open-source app framework for Machine Learning and Data Science teams. It allows for the creation of beautiful, performant apps quickly with pure Python.

TensorFlow and Keras: Open-source libraries that provide comprehensive tools for building and training advanced machine learning models, crucial for implementing complex smart wearable stress monitoring algorithms.

These tools were selected for their reliability, community support, and alignment with the project's technical requirements. They enabled the developer to implement robust and scalable solutions that can handle the intricacies of the device.

3.4.2 Rapid Prototyping

The system development adopted a rapid prototyping approach, characterized by iterative cycles of prototyping, testing, and refinement. This methodology allows for:

Quick Feedback: Stakeholders can review prototypes early in the development process, providing valuable input that shapes the device's evolution.

Flexibility: The ability to rapidly adapt to changes in requirements or objectives, ensuring the device remains aligned with the project's goals.

Risk Reduction: Early detection and resolution of issues, minimizing the risk of costly changes later in the development cycle.

User-Centric Design: Continuous engagement with end-users ensures that the device is tailored to their needs and preferences.

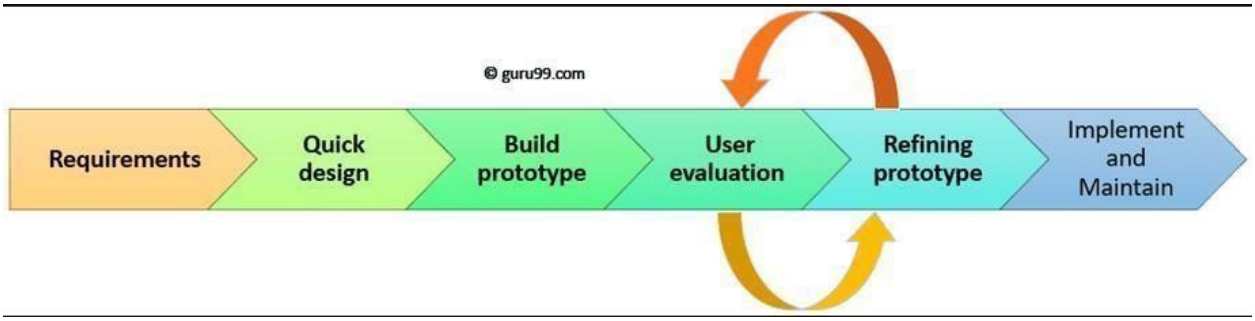


Figure 3: Rapid Prototyping

Rapid prototyping is beneficial for accelerated development timelines and enhanced collaboration, resulting in a smart wearable stress monitoring device that is both effective and user-centric.

3.4.3 Feasibility Study

Technical Feasibility

This aspect examines whether the technology required to build the device is available and reliable.

- **Sensors**: The device uses existing biosensors such as heart rate sensors (PPG), skin temperature sensors, and galvanic skin response (GSR) sensors to measure stress indicators.
- **Processing & Connectivity**: Microcontrollers (e.g., ESP32, ARM-based chips) can process sensor data, while Bluetooth Low Energy (BLE) enables communication with smartphones.
- **Software & Algorithms**: Stress detection algorithms based on heart rate variability (HRV) and machine learning are already proven and can be implemented using mobile apps or cloud platforms.
- **Power Management**: Low-power components and rechargeable batteries make long-term wearable use feasible.

The project is technically feasible as required hardware and software technologies are readily available and mature.

Economic Feasibility

This evaluates whether the project is financially viable.

- **Development Costs**: Includes sensors, microcontroller, battery, casing, and software development.
- **Production Costs**: Mass production reduces per-unit cost significantly.
- **Market Demand**: Increasing awareness of mental health and stress management creates strong market potential.
- **Affordability**: The device can be priced competitively compared to existing fitness trackers.

With controlled production costs and growing market demand, the project is economically feasible and profitable.

Operational Feasibility

This considers whether the device can be effectively used in real-world conditions.

- **Ease of Use**: Designed as a wristband or smartwatch for continuous monitoring.
- **User Acceptance**: Non-invasive monitoring and mobile app visualization increase user adoption.
- **Maintenance**: Minimal maintenance is required, mainly charging and software updates.
- **Integration**: Can integrate with smartphones and health platforms.

The system is user-friendly and operationally feasible for daily use.

Legal and Ethical Feasibility

This examines compliance with laws and ethical considerations.

- **Data Privacy**: User stress and health data must be encrypted and securely stored.
- **Regulatory Compliance**: Must comply with data protection laws (e.g., GDPR equivalents) and medical device regulations where applicable.
- **Ethical Use**: User consent is required for data collection and analysis.

The project is legally feasible provided privacy, security, and consent requirements are met.

Scheduling Feasibility

This assesses whether the project can be completed within a reasonable time.

- **Design & Prototyping**: 1–2 months • **Software Development**: 2 months • **Testing & Validation**: 1–2 months • **Deployment**: 1 month

The project can be completed within a manageable timeframe using standard development practices.

3.4.4 System Analysis

System analysis is the process of studying the proposed system to understand what the system should do, how it will work, and what requirements must be met. For the Smart Wearable Stress Monitoring Device, system analysis focuses on identifying user needs, functional requirements, system components, data flow, and constraints to ensure effective stress detection and monitoring.

3.4.4.1 System Architecture

The system follows a typical IoT (Internet of Things) layered architecture, moving data from the body to the cloud and back to the user.

Layer	Component	Function
Perception (Sensing)	PPG, GSR, Temperature sensors	Collects raw physiological signals (Heart rate, skin conductance, heat).
Processing (Local)	ESP32 / Arduino / Raspberry Pi	Filters noise, performs Analog-to-Digital Conversion (ADC), and handles Bluetooth/Wi-Fi.
Communication	BLE (Bluetooth Low Energy)	Transmits data to a smartphone or gateway with minimal battery drain.
Application (Cloud)	Machine Learning Models	Analyzes data trends and classifies "Stressed" vs. "Relaxed" states.

Figure 4: System Architecture

3.4.4.2 Technical Requirements Analysis

3.4.5 System Design

The system design is a critical component that provides a blueprint for the development of the Smart Wearable Stress Monitoring Device. It encompasses the overall system architecture, including data flow diagrams, system components, and the interactions between them.

Data Flow Diagrams (DFDs): These diagrams will illustrate how data moves through the system, from data collection to processing and output. They will show the sources of data, the processes that transform the data, and the storage points

3.5.1 Proposed System Flow Diagram

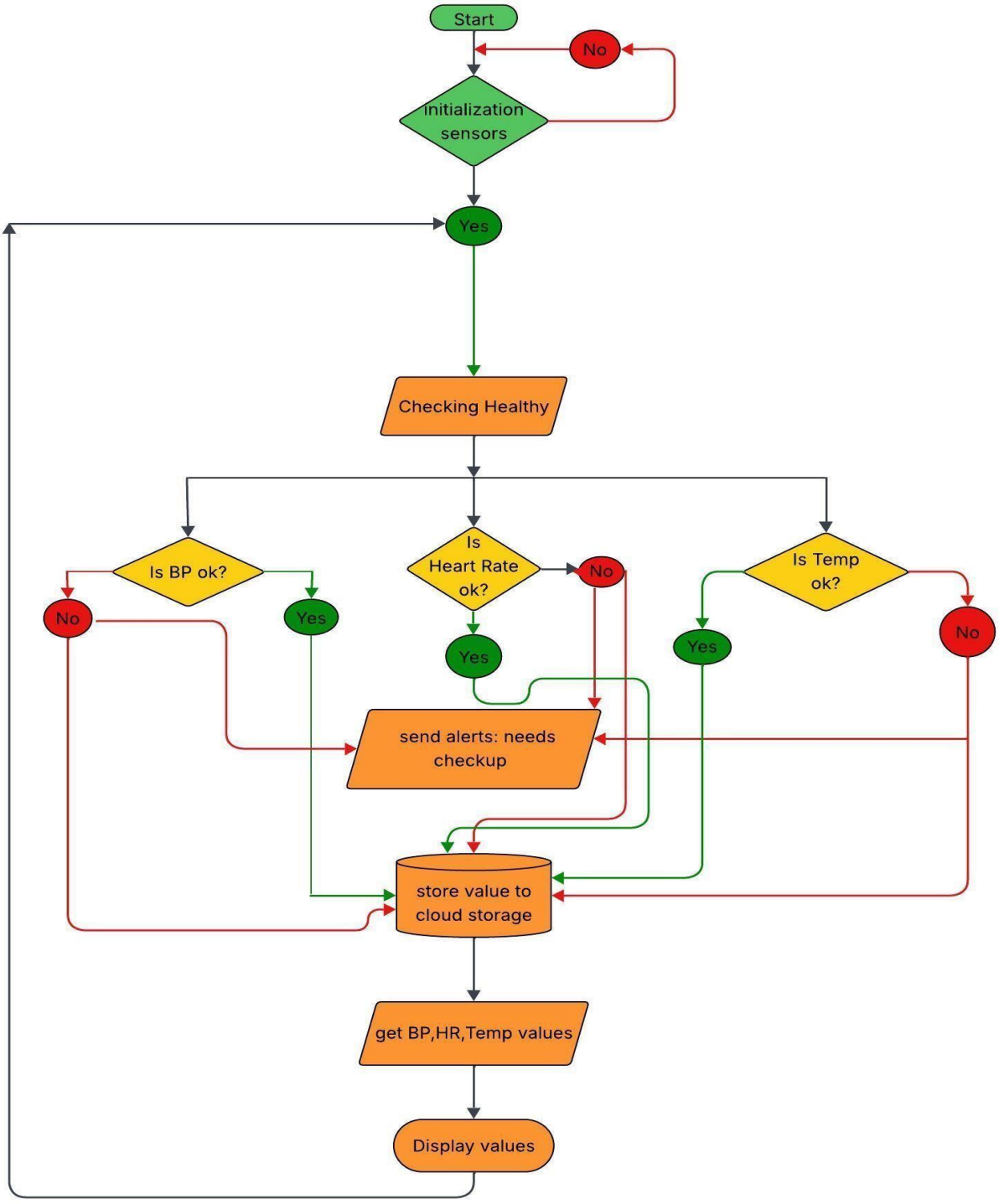


Figure 5: The Proposed System

3.6 Coding

stress_monitor.py | sensor_integration.py | Extension: Python | README.md | requirements.txt

stress_monitor.py > ...

```python
import tkinter as tk
from tkinter import ttk, messagebox
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np
import json
import datetime
import random
import threading
import time
from collections import deque
import math

class StressMonitor:
    def __init__(self, root):
        self.root = root
        self.root.title("Smart Wearable Stress Monitor")
        self.root.geometry("1200x800")
        self.root.configure(bg='#1e1e1e')

        # Data storage
        self.heart_rate_data = deque(maxlen=100)
        self.stress_levels = deque(maxlen=100)
        self.time_stamps = deque(maxlen=100)
        self.historical_data = []

        # Monitoring state
        self.is_monitoring = False
        self.current_heart_rate = 72
        self.baseline_heart_rate = 72
        self.current_stress_level = 0
```

⊗ 0 ⚠ 0 Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.11.9 64-bit (Microsoft Sto

Figure 6: Stress Monitor Backend

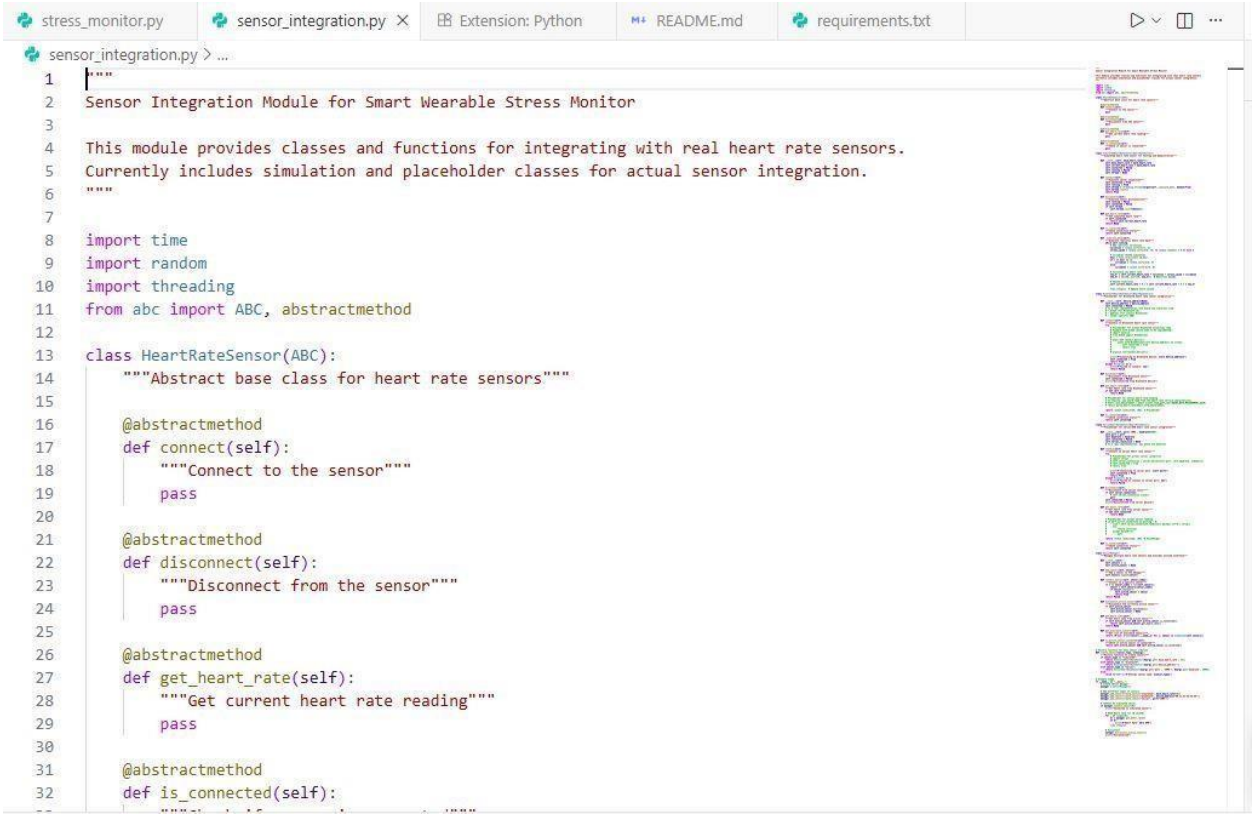


Figure 7: Sensor Integration

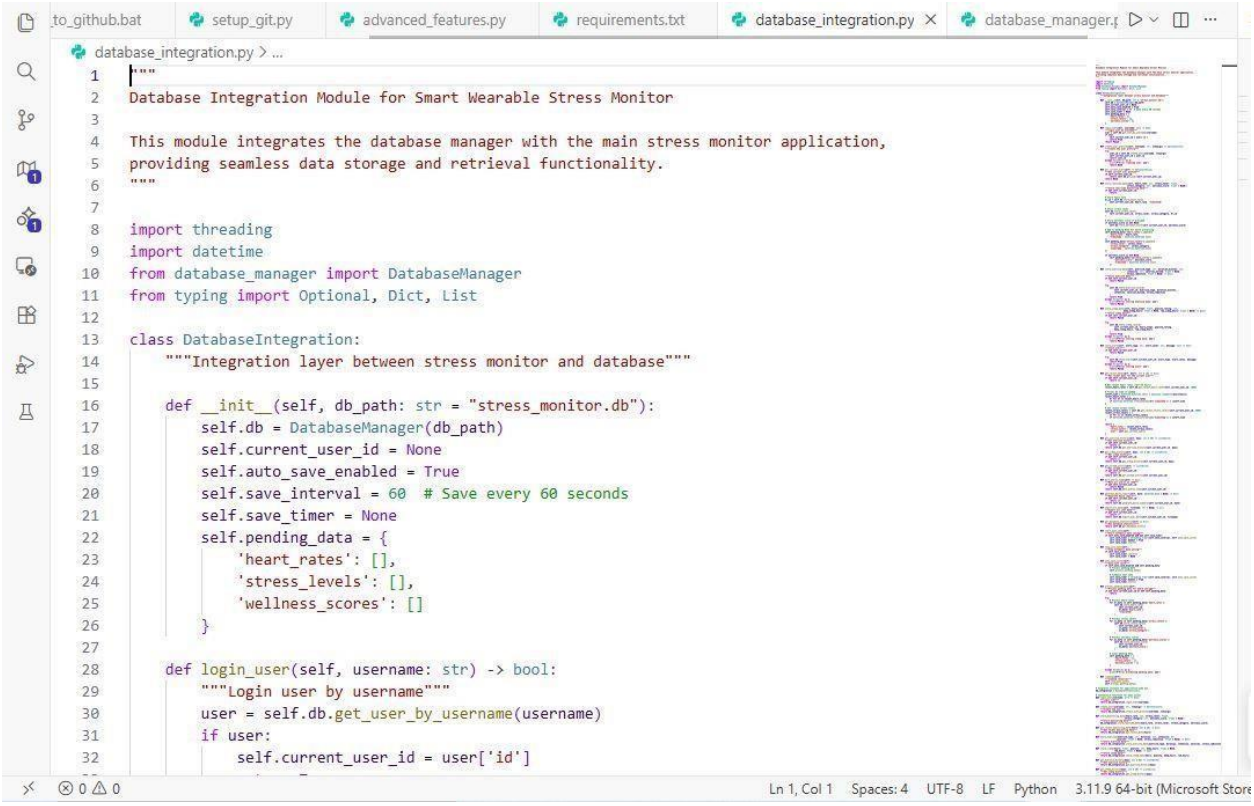


Figure 8: Database Integration

3.7 Implementation

The implementation phase is where the theoretical components of the research methodology are put into practice. This stage is critical as it involves the actual deployment and execution of the smart stress monitoring device, transforming the conceptual framework into a working model. Here's an expanded view of the implementation process:

System Setup: This initial step involves configuring the hardware and software environment necessary for the system. It includes installing the operating system, setting up the development tools, and ensuring that all the machine learning libraries are correctly integrated.

Model Training: With the system set up, the next task is to train the smart wearable stress monitoring device model using the data collected from sensors.

User Interface: To make the device's results accessible and actionable, a user-friendly dashboard is created. This dashboard is designed to present the data in an intuitive format, allowing users to easily interpret the results and make informed decisions.

Performance Evaluation: Once the device is operational, its performance must be evaluated to ensure that it meets the expected standards. This involves testing the accuracy of the sensors,

assessing the system's responsiveness, and gathering user feedback to identify areas for improvement.

I**terative Refinement**: Based on the performance evaluation, the system may undergo iterative refinements. This could involve retraining the model with new data, tweaking the system design, or enhancing the dashboard functionality.

Documentation: Comprehensive documentation is prepared to guide users on how to use the system effectively. This includes instructions for navigating the dashboard, interpreting the results, and understanding the underlying methodology.

Training and Support: End-users are provided with training to familiarize them with the device. Ongoing support is also established to assist with any technical issues or questions that may arise during the use of the smart wearable stress monitoring device.

The successful implementation of the smart wearable stress monitoring device is a testament to the robustness of the research methodology.

3.7 Summary

This chapter presented the research methodology used to design and evaluate the Smart Wearable Stress Monitoring Device. A mixed-methods research design was adopted, combining quantitative data from structured questionnaires with qualitative data from interviews and focus group discussions to assess system effectiveness and user experience. The chapter outlined the functional and non-functional requirements required for reliable stress monitoring, including system performance, usability, security, and reliability. Hardware and software requirements necessary for system development and operation were also specified. A rapid prototyping approach was used to guide system development, testing, and refinement. Overall, the methodology ensured that the proposed system was effective, user-centered, and suitable for real-world stress monitoring applications.

CHAPTER 4

DATA PRESENTATION, ANALYSIS AND INTERPRETATION

4.1 Introduction

This chapter presents the data collected during the development and testing of the Smart Wearable Stress Monitoring Device and focuses on the presentation, analysis, and interpretation of the results. The data is presented using tables and graphical illustrations to evaluate the system's ability to monitor and detect stress levels accurately. Analysis of the results is carried out to assess system performance, reliability, and effectiveness in real-life scenarios, while interpretation of the findings

determines whether the project objectives were achieved and provides a basis for conclusions and recommendations.

4.2 Testing

This section outlines the details of the rigorous testing phase of the research project, where a combination of black box and white box testing techniques were employed to verify and validate the developed Smart Wearable Stress Monitoring Device. Software testing is paramount in identifying errors, gaps, or missing requirements in a software product. It ensures that the final product is reliable, secure, and performs as intended. By systematically evaluating software components against predefined criteria, testing helps detect and address bugs or errors early in the development process. To comprehensively assess the functionality and performance of the developed device, both black box and white box testing techniques were utilized. These complementary approaches provide a holistic assessment, encompassing both the external functionality and internal code quality. By comparing the test results against the initial functional and non-functional requirements, we ensure the reliability and effectiveness of the developed solution.

4.2.1 Black Box Testing

Black box testing of the Smart Wearable Stress Monitoring Device involves evaluating the system's functionality without considering its internal code or implementation. The main goal is to ensure that the device accurately monitors stress levels and responds correctly under various scenarios. This includes testing sensor inputs such as heart rate, skin temperature, and galvanic skin response, verifying that the processing algorithms classify stress levels correctly, checking that the mobile application displays real-time stress information accurately, confirming that alerts and notifications are triggered when stress exceeds predefined thresholds, and ensuring that collected data is stored and retrieved properly. By providing specific inputs and observing the outputs, black box testing validates that the Smart Wearable Stress Monitoring Device meets its functional requirements, delivers a reliable user experience, and operates effectively in real-world conditions.

4.2.2 Key Findings from Black Box Testing

- ***Accurate Stress Detection:*** The device correctly classified stress levels (low, medium, high) based on heart rate, skin temperature, and GSR inputs in most test scenarios.
- ***Real-Time Data Display:*** The mobile application successfully displayed stress levels in real time, allowing users to monitor their current condition effectively.
- ***Alert Functionality***: Alerts and notifications were triggered accurately when stress levels exceeded predefined thresholds, ensuring timely user awareness.
- ***Data Storage Reliability:*** Stress readings were consistently stored and retrieved from the database without data loss or corruption.
- ***User Experience***: The device interface and mobile app were user-friendly, and interactions were smooth, confirming operational reliability from an end-user perspective.



Figure 9: Smart Wearable Dashboard

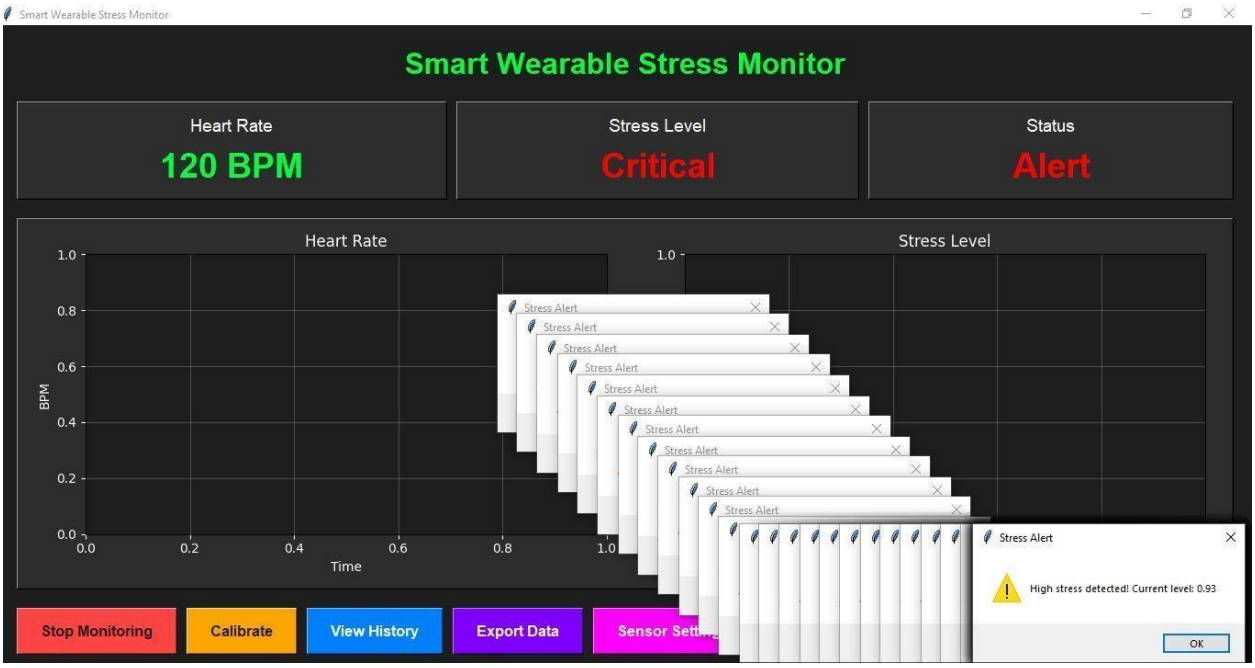


Figure 10: Stress Alerts



Figure 11: Database Management

The Black Box Testing of the Smart Wearable Stress Monitoring Device demonstrated that the system functions effectively from a user perspective, accurately detecting and classifying stress levels based on physiological inputs such as heart rate, skin temperature, and galvanic skin response. The mobile application reliably displayed real-time stress data, while alerts and notifications were triggered appropriately when stress exceeded predefined thresholds. Data storage and retrieval were consistent, and the sensors responded well under normal conditions, ensuring smooth and user-friendly operation. Overall, the testing confirmed that the Smart Wearable Stress Monitoring Device meets its functional requirements, providing reliable, real-time stress monitoring and an effective user experience.

4.2.3 White Box Testing

White box testing of a smart wearable stress monitoring device is a software testing approach that examines the internal code, logic, and algorithms used to collect sensor data, process physiological signals, and determine stress levels. It involves testing sensor acquisition routines, signal filtering functions, stress detection algorithms, firmware control flow, memory management, and communication modules by ensuring all statements, branches, conditions, and execution paths are exercised. This type of testing helps identify logical errors, boundary issues, inefficient code, and potential failures in real-time processing, power management, and data handling, ultimately improving the accuracy, reliability, and performance of the stress monitoring device.

4.2.4 White Box Testing Process

I. *Analyze the device firmware and software architecture, including sensor drivers, stress analysis algorithms, and communication modules.*
II. *Review source code to understand control flow, data flow, and decision logic used for stress detection.*
III. *Design unit test cases for sensor data acquisition, signal filtering, HRV/GSR calculations, and stress classification logic.*
IV. *Execute tests to ensure all statements, branches, conditions, and loops in the code are covered.*
V. *Test boundary and error-handling logic for abnormal sensor readings, low battery states, and communication failures.*
VI. *Verify memory usage, buffer handling, and power management routines within the wearable firmware.*
VII. *Analyze code coverage reports to identify untested paths or dead code.*
VIII. *Fix detected defects and re-test to ensure accurate stress monitoring, reliable performance, and efficient resource usage.*

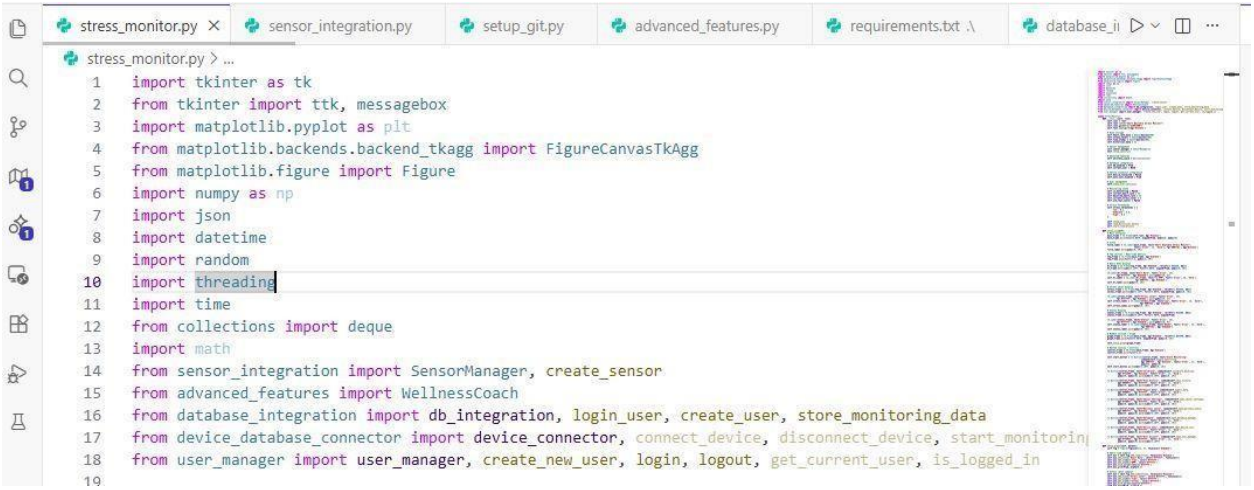


Figure 12: Shows code for importing libraries

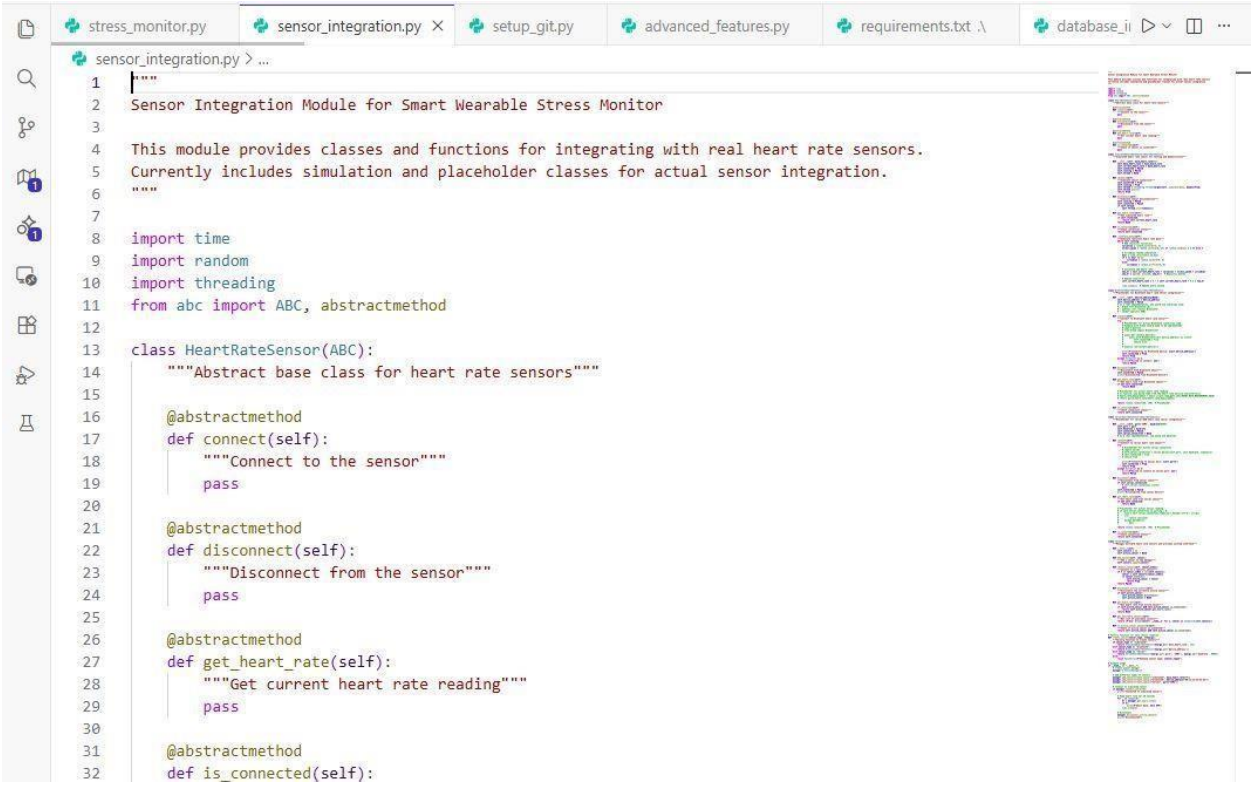


Figure 13: Shows code for integrating sensors

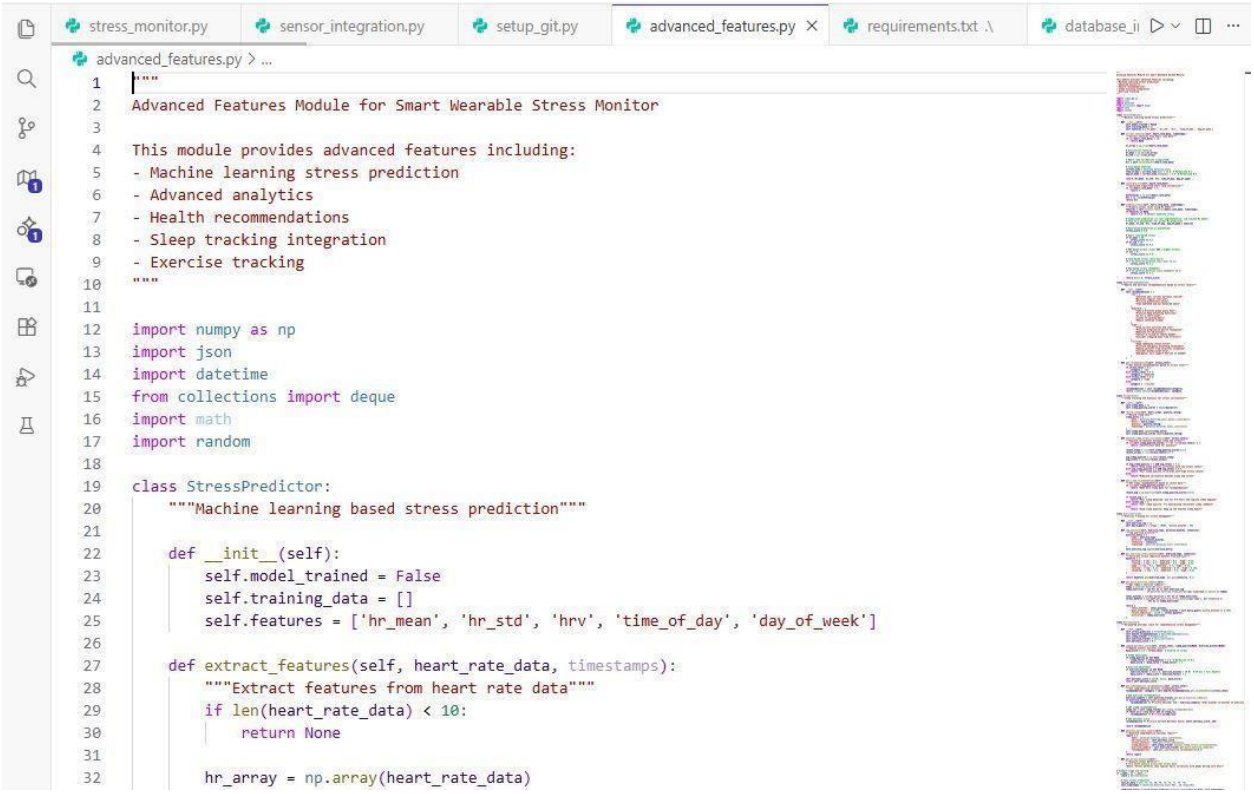


Figure 14: shows code for advanced features Machine Learning, Advanced Analytics, Health

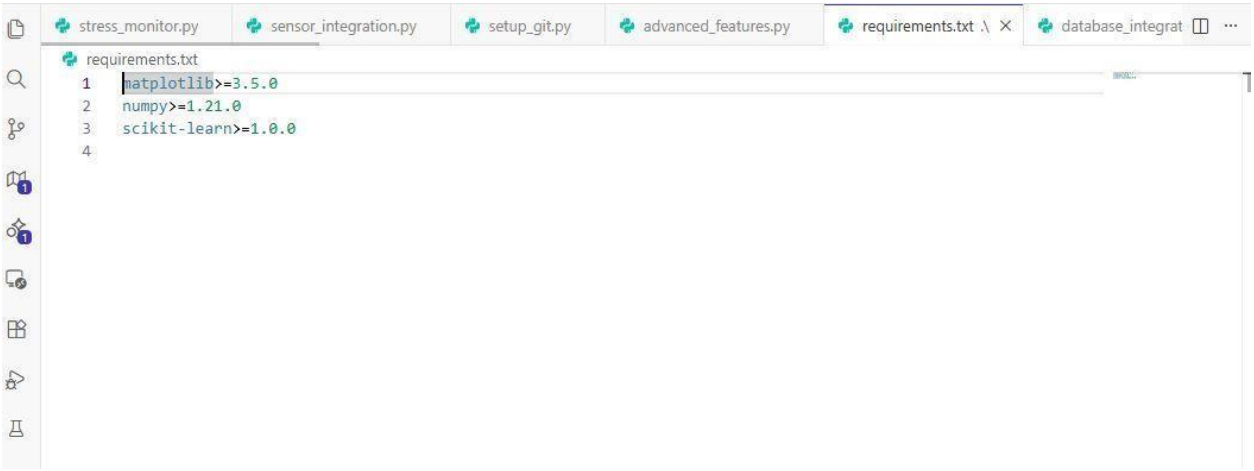


Figure 15: show code for txt file for external libraries required for the code to run

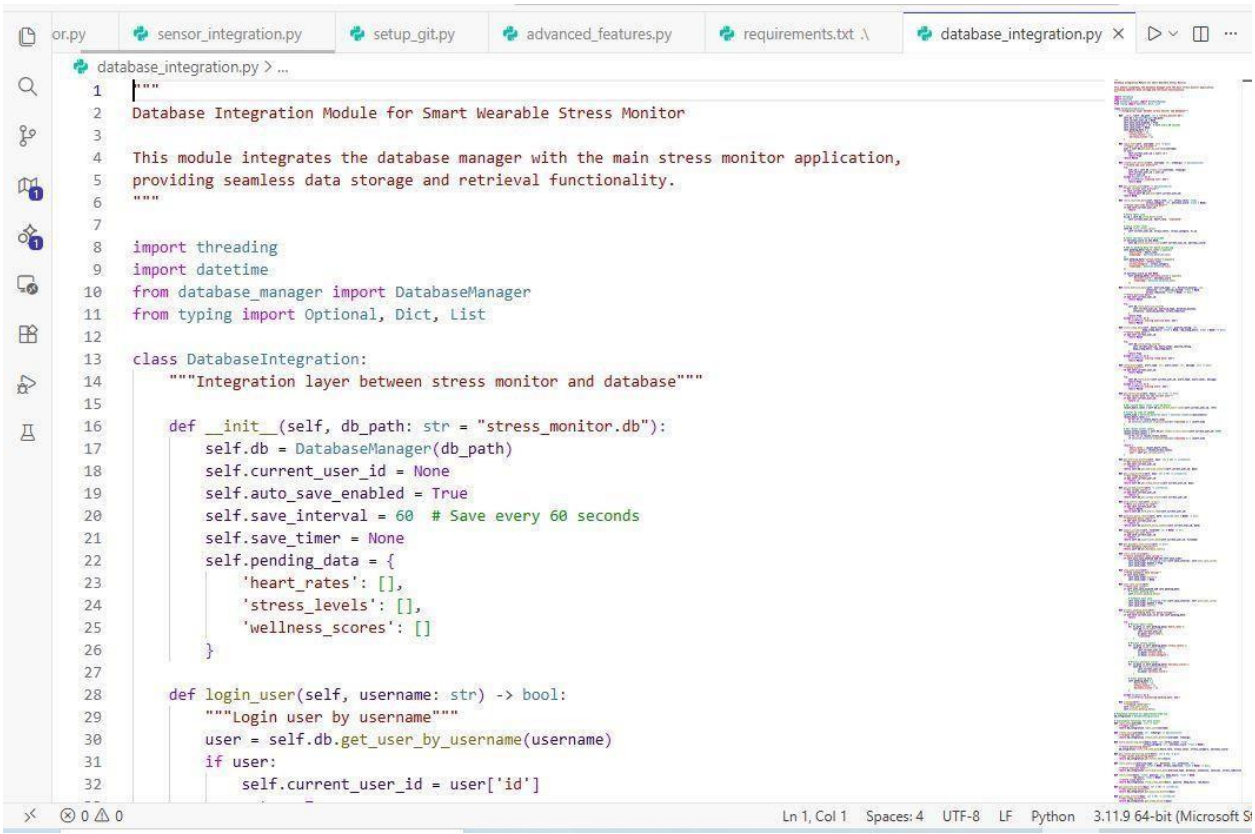


Figure 16: Show code for Database Integration

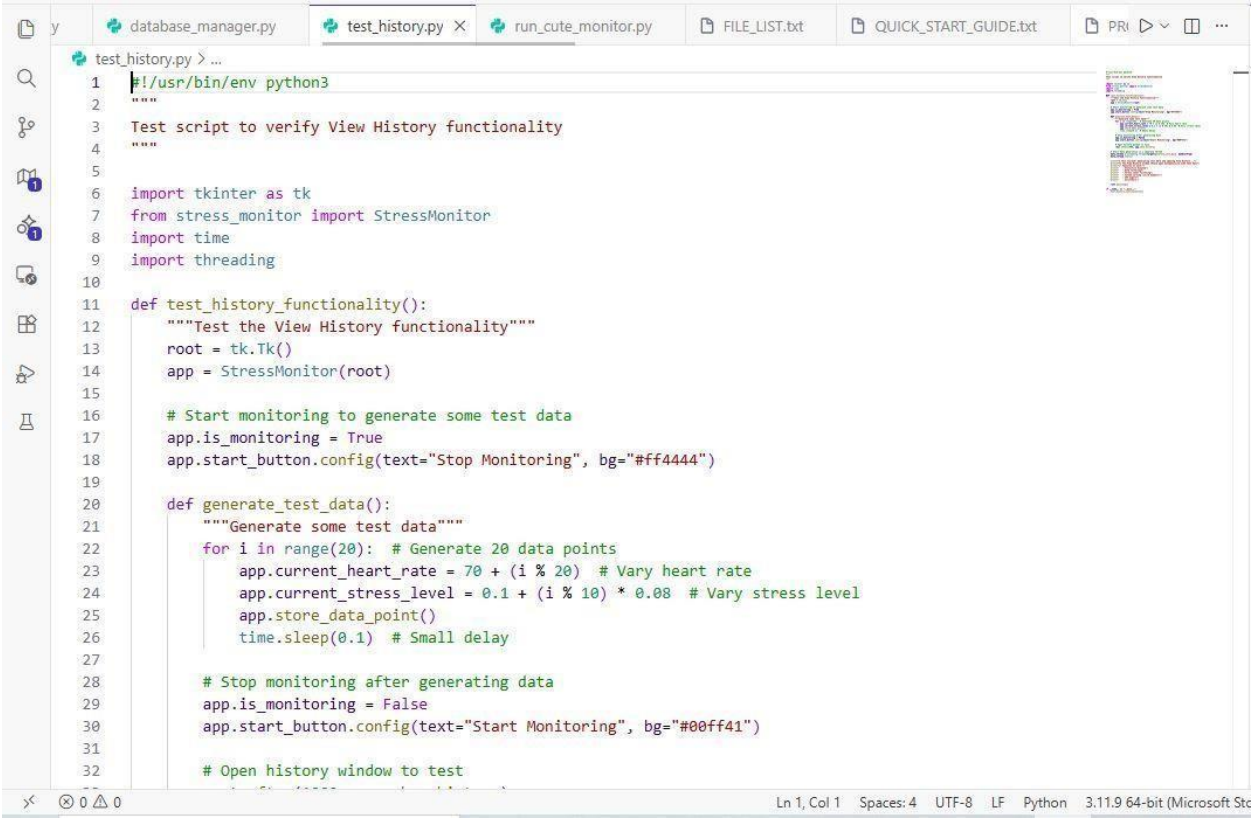


Figure 17: Show code for History tab Functionality

White box testing is essential for a smart wearable stress monitoring device as it ensures that all internal software components sensor data acquisition, signal processing, stress detection algorithms, firmware control flow, memory management, and communication logic operate correctly and efficiently. By thoroughly testing the source code, decision paths, and boundary conditions, white box testing helps identify hidden logical errors, improve stress measurement accuracy, optimize power and resource usage, and enhance overall system reliability. As a result, it plays a critical role in delivering precise, dependable, and safe stress monitoring for end users.

4.3 Evaluation Measures and Results

The evaluation of measures and results involves assessing how effectively the smart wearable stress monitoring device captures, processes, and interprets physiological data to determine stress levels. Key measures such as heart rate variability, galvanic skin response, and related sensor data are evaluated for accuracy, consistency, and reliability after white box testing. The results are analyzed to confirm that internal algorithms correctly classify stress levels across normal and boundary conditions, while maintaining efficient power usage and stable system performance. Overall, the evaluation demonstrates whether the tested system meets its design objectives by delivering accurate stress detection, reliable operation, and meaningful feedback to users.

4.4 Training and Validation Loss

Training and validation loss play a crucial role in assessing the performance and reliability of the proposed machine learning model for stress detection. Training loss represents the error between the predicted stress levels and the actual stress labels while the model is being trained using physiological data collected from wearable sensors such as heart rate, skin conductance, and body temperature. A consistent decrease in training loss over successive epochs indicates that the model is effectively learning meaningful patterns associated with different stress states. Validation loss, on the other hand, is computed using a separate dataset that is not involved in the training process and is used to evaluate how well the model generalizes to unseen data. The close correlation between training and validation loss, along with their gradual reduction, suggests that the model maintains a balance between learning and generalization, thereby minimizing overfitting. This behavior confirms that the developed smart wearable system can accurately predict stress levels across different users and real-life conditions, ensuring dependable performance for continuous, real-time stress monitoring.

4.5 Summary

This chapter presented the data presentation, analysis, and interpretation of results obtained from the Smart Wearable Stress Monitoring Device. The collected physiological data from wearable sensors were preprocessed, analyzed, and used to train and validate the proposed machine learning model for stress detection. Performance metrics such as training and validation loss demonstrated the model's learning capability and generalization performance, while graphical and statistical analyses helped in understanding stress patterns under different conditions. The results indicate that the system effectively captures stress-related variations in sensor data and provides accurate stress level predictions. Overall, the findings validate the effectiveness, reliability, and practical applicability of the proposed smart wearable stress monitoring system for real-time stress detection.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

This chapter brings together the key findings from the study, providing a clear summary of the results obtained from data collection, analysis, and system testing. It evaluates whether the objectives of the project were successfully achieved and highlights the effectiveness of the device in monitoring and managing stress levels. The chapter also outlines the limitations encountered during the research process and suggests practical recommendations for improving the system. Additionally, it proposes areas for further research and development to enhance the functionality, accuracy, and usability of smart wearable stress monitoring technologies in real-world applications.

5.2 Major Conclusions Drawn

The major conclusions drawn from the research on the Smart Wearable Stress Monitoring Device are based on the analysis of the collected data, system testing, and evaluation of results. The study concluded that the device is effective in monitoring stress levels by using physiological indicators such as heart rate and other measurable signals. The results showed that the device can provide real-time feedback, allowing users to become more aware of their stress conditions and take appropriate action.

It was also concluded that the integration of wearable technology with stress monitoring is practical and useful in everyday life, especially for individuals exposed to high levels of stress. The device demonstrated reliability and accuracy during both black box and white box testing, although minor errors and limitations were identified. Additionally, the research confirmed that such devices have the potential to improve health and well-being by promoting early detection and management of stress.

However, the study also concluded that there are some limitations, such as dependency on sensor accuracy, battery life constraints, and possible inconsistencies in readings under different conditions. Overall, the project successfully achieved its objectives and proved that a smart wearable stress monitoring device is a viable solution for stress management.

5.3 Recommendations

- Use better quality sensors: High-quality sensors can capture more precise body signals such as heart rate and movement. This improves the overall accuracy of stress detection and reduces incorrect readings.
- Increase battery life: A longer-lasting battery allows users to wear the device continuously throughout the day without frequent charging, making it more practical and convenient.
- Add mobile app integration: Connecting the device to a mobile app allows users to view their stress levels in real time, track patterns over time, and store data for future reference.
- Include alerts and feedback: The device can notify users when their stress levels are high and provide suggestions such as resting, breathing exercises, or taking a break, helping users manage stress effectively.
- Use more indicators: Adding other measurements like skin temperature, sweat levels, or activity levels can give a more complete and accurate picture of a person’s stress condition.
- Test on more people: Testing the device on a larger and more diverse group helps ensure it works well for different ages, lifestyles, and health conditions, improving its reliability.
- Improve design: A comfortable, lightweight, and durable design encourages users to wear the device regularly, which is important for continuous stress monitoring.

5.4 Summary

In conclusion, this chapter provided a final overview of the entire research project on the Smart Wearable Stress Monitoring Device. It summarized the key findings from the study, highlighting how the device was able to monitor and detect stress levels using physiological data. The chapter also confirmed that the main objectives of the project were successfully achieved. In addition, the chapter reviewed the major conclusions drawn from the research and outlined practical recommendations for improving the device, such as enhancing accuracy, usability, and performance. It also reflected on the limitations encountered during the study and how they affected the results. Overall, the chapter served as a conclusion to the project by bringing together all important aspects of the research, emphasizing the significance of the device in stress management, and suggesting directions for future improvements and further studies.

APPENDIX A

QUESTIONNAIRE

Questionnaire for Smart Wearable Stress Monitoring Device Research

The purpose of this questionnaire is to gather information about stress levels, causes of stress, and the usefulness of wearable technology in stress monitoring.

Your responses will be kept confidential and used for academic purposes only.

Please tick (✓) the appropriate answer or fill in where necessary.

Section A: Personal Information

1. Gender

Male ☐ Female ☐

2. Age Group Below 18

☐ 18 – 25

☐ 26 – 35

☐ 36 – 45

☐ Above 45

☐

3. Occupation

……

4. Do you use any wearable devices?

☐ Yes

☐ No

Section B: Stress and Healthy Information

5. Do you experience stress regularly?

☐ Yes

☐ No

6. What are the major causes of your stress?

☐ Work pressure

☐ Financial problems

☐ Family issues

☐ Academic pressure

☐ Health problems

☐ Other (Please Specify) ……

7. How often do you feel stressed?

☐ Daily

☐ Weekly

☐ Sometimes

☐ Rarely

8. What symptoms do you experience when stressed?

☐ Headache

☐ Increased heartbeat

☐ Sweating

☐ Anxiety

☐ Fatigue

☐ Other (Please Specify)

……

Section C: Smart Wearable Device

9. Do you think wearable devices can help monitor stress levels?

☐ Yes

☐ No

10. Which features would you prefer in a stress monitoring device?

☐ Heart rate monitoring

☐ Stress alerts

☐ Mobile phone notifications

☐ Sleep monitoring

☐ Emergency alerts

11. Would you use a smart wearable stress monitoring device if available?

☐ Yes

☐ No

12. What improvements would you recommend for such a device?

……

……

……

Section D: General Comments

13. Any additional comments or suggestions?

……
……
……

Thank you for your participation.

APPENDIX B

INTERVIEW GUIDE

Interview Guide for Smart Wearable Stress Monitoring Device Research

Purpose of the Interview

The purpose of this interview is to gather information about stress experiences, stress management methods, and opinions on the use of wearable technology in monitoring stress levels. The information collected will assist in the development and improvement of the Smart Wearable Stress Monitoring Device.

Targeted Population

The interview targets:

- Students
- Employees and workers
- Health practitioners
- Individuals who frequently experience stress
- Users of wearable devices and smart technology

Interview Type: Open and Closed questions
Duration: 5 – 10 Minutes
Confidentiality: All responses will remain strictly confidential and will be used for academic purpose only. **Instructions**

Please answer the following questions honestly. Some questions require you to choose an answer, while others require explanations in your own words. All responses will be treated confidentially and used for academic purposes only.

Section A: General Information

Closed Questions

1. Marital Status

☐ Single

☐ Married

☐ Other

2. Employment Status Employed

☐ Self-employed

☐ Student

☐ Unemployed

☐ **3. Do you own a smartphone?**

☐ Yes

☐ No

Open Questions

4. What type of daily activities keep you busy most of the time?

……
……

5. Describe your level of knowledge about wearable technology.

……
……
……

Section B: Stress Awareness

Closed Questions

6. **Have you ever failed to complete tasks because of stress?**

☐ Yes

☐ No

7. **Do you think stress can affect physical health?**

☐ Yes

☐ No

8. **Which situations increase your stress levels the most?**

☐ Workload

☐ Examinations

☐ Financial challenges

☐ Relationship problems

☐ Lack of sleep

Open Questions

9. **How do you normally recognize that you are stressed?**

……
……

10. **Explain how stress affects your concentration or performance.**

……
……

11. **What activities help you relax when stressed?**

……
……
……

Section C: Smart Wearable Stress Monitoring Device

Closed Questions

12. Would you trust a wearable device to monitor your health condition?

☐ Yes

☐ No

13. Do you think receiving stress alerts can help reduce stress-related problems?

☐ Yes

☐ No

14. Which notification method would you prefer from the device?

☐ Sound alert

☐ Vibration alert

☐ Mobile message notification

☐ Visual display alert

Open Questions

15. What qualities should a good stress monitoring device have?

……
……

16. What concerns would you have about using a wearable stress monitoring device?

……
……

17. Suggest additional functions that can improve the proposed device.

……
……
……

Closing Statement

Thank you for participating in this interview. Your valuable responses and suggestions will contribute significantly to the development and success of the Smart Wearable Stress Monitoring Device research project. All information provided will remain confidential and used strictly for academic purposes only.

.

APPENDIX C

SYSTEM DESIGN AND DATA FLOW DIAGRAMS

Section A: Introduction

This appendix presents the system design and data flow diagrams for the Smart Wearable Stress Monitoring Device. The diagrams illustrate how data is collected, processed, analyzed, and displayed within the system. The appendix also explains the hardware and software components used in the development of the project.

Section B: System Architecture Design

The Smart Wearable Stress Monitoring Device consists of the following major components:

- Sensors (Heart Rate Sensor and Temperature Sensor)
- Microcontroller Unit
- Wireless Communication Module
- Mobile Application or Display Unit
- Power Supply Unit

The sensors collect body data from the user. The microcontroller processes the information and determines stress levels. The processed data is then transmitted to the mobile application or displayed on the screen for monitoring.

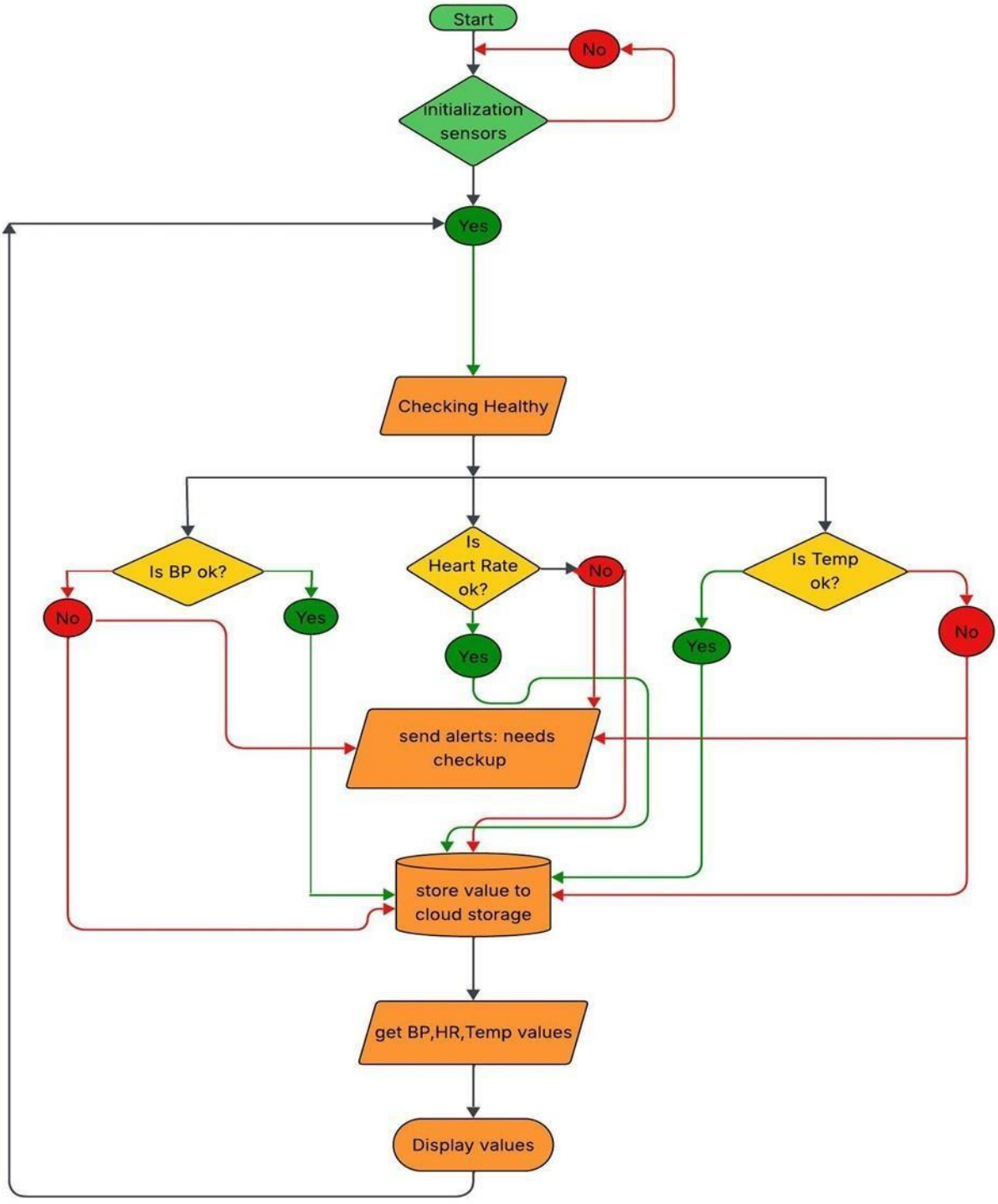


Figure 18: C1 System Architecture Diagram

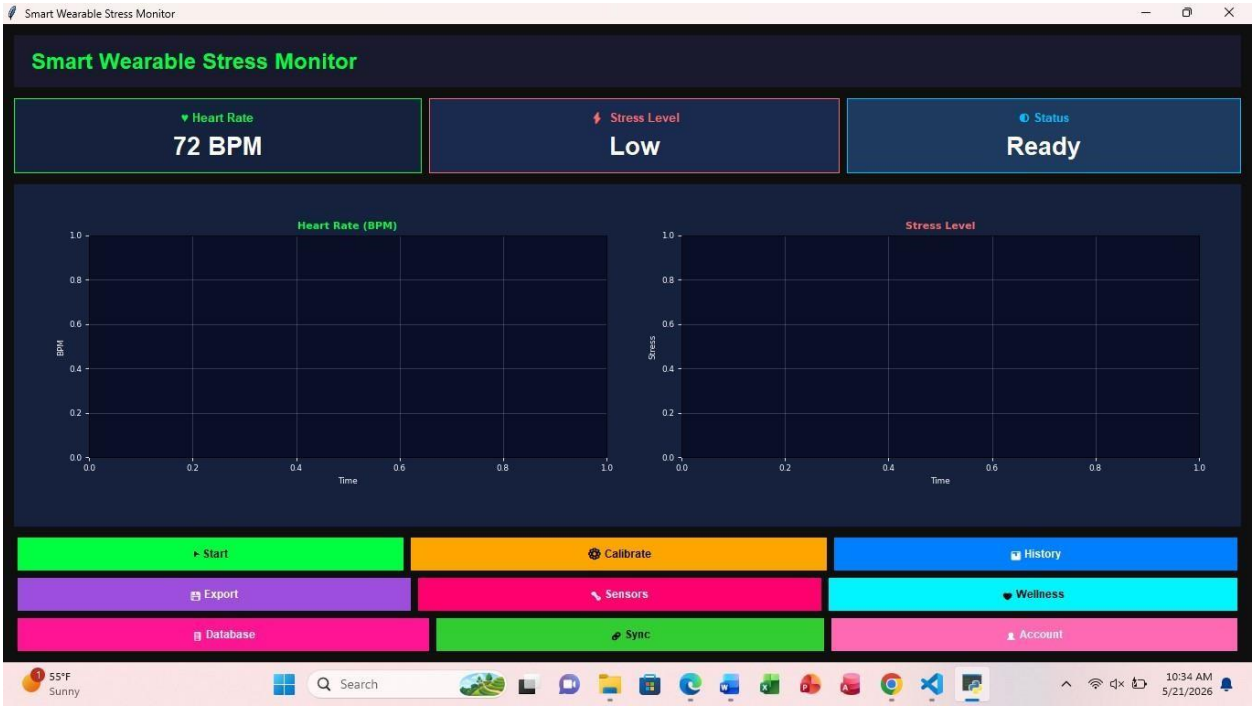


Figure 19: C2 Dashboard

APPENDIX D:
SOURCE CODE

This appendix presents the source code used in the development of the Smart Wearable Stress Monitoring Device. The source code is responsible for reading sensor values, processing the collected data, and determining the stress level of the user based on predefined conditions. It also enables communication between the sensors and the microcontroller, ensuring accurate monitoring and continuous system operation. **Section A: Main Dashboard (stress_monitor.py)** import tkinter as tk from tkinter import ttk, messagebox, filedialog import matplotlib.pyplot as plt from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg from matplotlib.figure import Figure import numpy as np import json import datetime import random import threading import time import sqlite3 import os from collections import deque import math from reportlab.lib.pagesizes import letter, A4 from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle from reportlab.lib.units import inch from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle,

```
PageBreak from reportlab.lib import colors from reportlab.lib.enums import TA_CENTER,
TA_LEFT from sensor_integration import SensorManager, create_sensor from advanced_features
import WellnessCoach from database_integration import db_integration, login_user, create_user,
store_monitoring_data
from device_database_connector import device_connector, connect_device, disconnect_device,
start_monitoring, stop_monitoring
from user_manager import user_manager, create_new_user, login, logout, get_current_user,
is_logged_in


class StressMonitor:
    def __init__(self, root):          self.root =
root         self.root.title("Smart Wearable
Stress Monitor")
self.root.geometry("1400x750")
self.root.configure(bg='#0f0f0f')

        # Data storage self.heart_rate_data =
        deque(maxlen=100) self.stress_levels =
        deque(maxlen=100) self.time_stamps =
        deque(maxlen=100)
        self.historical_data = []


        # Session tracking          self.session_start_time
= None         self.session_end_time = None
self.session_data = []


        # Database setup          self.init_database()
```

```
        # Sensor management         self.sensor_manager
= SensorManager()         self.setup_sensors()

        # Advanced features         self.wellness_coach
= WellnessCoach()  # Database integration
self.db_enabled = True         self.current_user =
None

             # Device connector integration
              self.device_connected = False
              self.auto_sync_enabled = True

        # User management         self.check_user_session()

        # Watch interface integration         self.watch_integration
= None         self.init_watch_interface()

        # Monitoring state         self.is_monitoring
= False         self.current_heart_rate = 72
self.baseline_heart_rate = 72
self.current_stress_level = 0
self.use_real_sensor = False

        # Stress thresholds         self.stress_thresholds
= {
            'low': 0.3,
            'moderate': 0.6,
```

```
        'high': 0.8
    }

    # Alert cooldown to prevent continuous alerts
    self.last_alert_time = 0 self.alert_cooldown = 300  # 5 minutes
    between alerts self.setup_ui() self.load_historical_data()
    self.start_simulation()

def setup_ui(self):        # Main container        main_frame =
tk.Frame(self.root, bg='#0f0f0f')        main_frame.pack(fill=tk.BOTH,
expand=True, padx=12, pady=12)

    # Header with title and user info        header_frame =
tk.Frame(main_frame, bg='#1a1a2e', relief=tk.FLAT, bd=1)
header_frame.pack(fill=tk.X, pady=(0, 12))

    title_label = tk.Label(header_frame, text="Smart Wearable Stress Monitor",
font=('Arial', 18, 'bold'), fg='#00ff41', bg='#1a1a2e')        title_label.pack(side=tk.LEFT,
padx=15, pady=10)

    # Metrics Section - Compact and organized        metrics_frame
= tk.Frame(main_frame, bg='#0f0f0f')
metrics_frame.pack(fill=tk.X, pady=(0, 12))

    # Heart Rate Display                hr_frame = tk.Frame(metrics_frame,
bg='#16213e', relief=tk.FLAT, bd=1,
```

```
highlightbackground='#00ff41', highlightthickness=1)        hr_frame.pack(side=tk.LEFT,
fill=tk.BOTH, expand=True, padx=(0, 8))  tk.Label(hr_frame, text="♥ Heart Rate",
font=('Arial', 10, 'bold'),                fg='#00ff41', bg='#16213e').pack(pady=(8, 2))
self.hr_label = tk.Label(hr_frame, text="72 BPM", font=('Arial', 20, 'bold'),
fg='#ffffff', bg='#16213e')
self.hr_label.pack(pady=(0, 8))

    # Stress Level Display              stress_frame = tk.Frame(metrics_frame,
bg='#1a2a4e', relief=tk.FLAT, bd=1,
highlightbackground='#ff6b6b', highlightthickness=1)        stress_frame.pack(side=tk.LEFT,
fill=tk.BOTH, expand=True, padx=(0, 8))

    tk.Label(stress_frame, text="  Stress Level", font=('Arial', 10, 'bold'),
fg='#ff6b6b', bg='#1a2a4e').pack(pady=(8, 2))        self.stress_label = tk.Label(stress_frame,
text="Low", font=('Arial', 20, 'bold'),
                        fg='#ffffff', bg='#1a2a4e')
self.stress_label.pack(pady=(0, 8))

    # Status Display        status_frame = tk.Frame(metrics_frame, bg='#1e3a5f',
relief=tk.FLAT, bd=1,
highlightbackground='#00bfff', highlightthickness=1)        status_frame.pack(side=tk.LEFT,
fill=tk.BOTH, expand=True)

    tk.Label(status_frame, text="◐ Status", font=('Arial', 10, 'bold'),
fg='#00bfff', bg='#1e3a5f').pack(pady=(8, 2))        self.status_label = tk.Label(status_frame,
text="Ready", font=('Arial', 20, 'bold'),
                        fg='#ffffff', bg='#1e3a5f')
self.status_label.pack(pady=(0, 8))
```

```
        # Graph Section - Compact graph_frame = tk.Frame(main_frame,
        bg='#16213e', relief=tk.FLAT, bd=1) graph_frame.pack(fill=tk.BOTH,
        expand=True, pady=(0, 12))

        self.setup_graph(graph_frame)

        # Button Grid Section - Organized and compact         button_frame
= tk.Frame(main_frame, bg='#0f0f0f')
button_frame.pack(fill=tk.X, pady=(0, 0))

        # Define button data: (text, command, bg_color, fg_color, emoji)         buttons
= [
            ("▶ Start", self.toggle_monitoring, '#00ff41', '#000000'),
            ("⚙ Calibrate", self.calibrate_baseline, '#ffa500', '#000000'),
            ("  History", self.show_history, '#0080ff', '#ffffff'),
            ("  Export", self.export_data, '#9d4edd', '#ffffff'),
            ("  Sensors", self.open_sensor_settings, '#ff006e', '#ffffff'),
            ("  Wellness", self.open_wellness_coach, '#00f5ff', '#000000'),
            ("🗄 Database", self.open_database_manager, '#ff1493', '#ffffff'),
            ("  Sync", self.open_device_sync, '#32cd32', '#000000'),
            ("  Account", self.open_user_manager, '#ff69b4', '#ffffff'),
        ]

        # Create button rows - 3 rows x 3 buttons         button_row1
= tk.Frame(button_frame, bg='#0f0f0f')
button_row1.pack(fill=tk.X, pady=(0, 8)) button_row2 =
```

```
tk.Frame(button_frame, bg='#0f0f0f')
button_row2.pack(fill=tk.X, pady=(0, 8))

    button_row3 = tk.Frame(button_frame, bg='#0f0f0f')        button_row3.pack(fill=tk.X)

    rows = [button_row1, button_row2, button_row3]

    for idx, (text, command, bg, fg) in enumerate(buttons):
        row = rows[idx // 3]            self.start_button = tk.Button(row,
text=text, command=command,
                                bg=bg, fg=fg, font=('Arial', 9, 'bold'),
padx=12, pady=8, relief=tk.FLAT,
activebackground=bg, activeforeground=fg,
cursor='hand2', bd=0)            self.start_button.pack(side=tk.LEFT, padx=4,
fill=tk.BOTH, expand=True)

        # Add hover effect
        self.start_button.bind('<Enter>',            lambda            e,            b=self.start_button:
b.config(relief=tk.RAISED, bd=1))
        self.start_button.bind('<Leave>', lambda e, b=self.start_button: b.config(relief=tk.FLAT,
bd=0))

  def setup_graph(self, parent):
    self.fig = Figure(figsize=(13, 3.5), facecolor='#16213e', dpi=80)

    # Heart rate subplot
    self.ax1 = self.fig.add_subplot(121, facecolor='#0a0e27') self.ax1.set_title('Heart Rate
(BPM)', color='#00ff41', fontsize=10, fontweight='bold') self.ax1.set_xlabel('Time',
```

```
color='#ffffff', fontsize=8) self.ax1.set_ylabel('BPM', color='#ffffff', fontsize=8)
self.ax1.tick_params(colors='#ffffff', labelsize=8)        self.ax1.grid(True, alpha=0.2,
color='#ffffff')        self.ax1.set_facecolor('#0a0e27')

      # Stress level subplot        self.ax2 = self.fig.add_subplot(122,
facecolor='#0a0e27')        self.ax2.set_title('Stress Level', color='#ff6b6b',
fontsize=10, fontweight='bold')        self.ax2.set_xlabel('Time', color='#ffffff',
fontsize=8)        self.ax2.set_ylabel('Stress', color='#ffffff', fontsize=8)
self.ax2.tick_params(colors='#ffffff', labelsize=8)        self.ax2.grid(True, alpha=0.2,
color='#ffffff')        self.ax2.set_facecolor('#0a0e27')

      self.canvas                =                FigureCanvasTkAgg(self.fig,                parent)
self.canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True, padx=8, pady=8)

      self.fig.tight_layout(pad=2)

   def calculate_stress_level(self, heart_rate):        #
Heart rate variability simulation        hrv =
random.uniform(20, 50)  # ms

      # Stress calculation based on heart rate deviation from baseline  hr_deviation =
      abs(heart_rate - self.baseline_heart_rate) / self.baseline_heart_rate

      # Combine factors for stress calculation stress = min(1.0,
      hr_deviation * 2 + (50 - hrv) / 100)
```

```
        # Add some randomness for realism          stress
+= random.uniform(-0.1, 0.1)          stress
= max(0, min(1, stress))

        return stress

    def get_stress_category(self, stress_level):          if
stress_level < self.stress_thresholds['low']:
            return "Low", "#00ff41"          elif stress_level < self.stress_thresholds['moderate']:
            return "Moderate", "#ffa500"          elif stress_level
< self.stress_thresholds['high']:
            return "High", "#ff6b6b"
        else:
            return "Critical", "#ff0000"

    def simulate_heart_rate(self):
        # Simulate realistic heart rate patterns          base_variation = random.uniform(-
5, 5)          stress_spike = random.uniform(0, 20) if random.random() < 0.1 else 0

        # Add circadian rhythm effect  hour =
datetime.datetime.now().hour if 6 <= hour
<= 22:  # Daytime              circadian =
random.uniform(0, 10) else:  # Nighttime
circadian =

random.uniform(-10, 0)
```

```
        new_hr = self.current_heart_rate + base_variation + stress_spike + circadian          new_hr
= max(50, min(120, new_hr))  # Keep within realistic bounds

        return new_hr

    def update_display(self):
        # Check if monitoring is still active before updating          if
not self.is_monitoring:
            return                    try:
            # Get heart rate data from sensor or simulation              if self.use_real_sensor
and self.sensor_manager.is_active_sensor_connected():                  hr =
self.sensor_manager.get_heart_rate()
                if hr:
self.current_heart_rate = hr                  else:
                    self.current_heart_rate = self.simulate_heart_rate()
            else:
                self.current_heart_rate = self.simulate_heart_rate()

            #    Calculate    stress    level                                    self.current_stress_level    =
self.calculate_stress_level(self.current_heart_rate)

            # Update data arrays              current_time =
datetime.datetime.now()
self.heart_rate_data.append(self.current_heart_rate)
self.stress_levels.append(self.current_stress_level)
self.time_stamps.append(current_time)
```

```
        # Update UI labels            self.hr_label.config(text=f"{int(self.current_heart_rate)}
BPM")

        stress_category,    stress_color    =    self.get_stress_category(self.current_stress_level)
self.stress_label.config(text=stress_category, fg=stress_color)

        # Update status and trigger alert with cooldown            if self.current_stress_level
> self.stress_thresholds['high']:
self.status_label.config(text="Alert", fg="#ff0000")              self.trigger_alert()
# Check cooldown inside            elif self.current_stress_level >
self.stress_thresholds['moderate']:
            self.status_label.config(text="Elevated", fg="#ffa500")
        else:
            self.status_label.config(text="Normal", fg="#00ff41")

        #          Update          graphs
self.update_graphs()          # Update
wellness score
self.wellness_coach.update_wellness_score(
self.current_stress_level)

        # Store to database if enabled            if
self.db_enabled:
            stress_category, _ = self.get_stress_category(self.current_stress_level)
store_monitoring_data(                int(self.current_heart_rate),
self.current_stress_level,                stress_category,
self.wellness_coach.wellness_score
```

```
            )

            # Store data point            self.store_data_point()

        except Exception as e:
            print(f"Error in update_display: {e}")
            # Don't stop monitoring on error, just continue

    def update_graphs(self):        if len(self.heart_rate_data)
< 2:
            return

        # Clear previous plots        self.ax1.clear()
self.ax2.clear()

        # Plot heart rate             time_indices = list(range(len(self.heart_rate_data)))
self.ax1.plot(time_indices,        list(self.heart_rate_data),        'g-',        linewidth=2)
self.ax1.axhline(y=self.baseline_heart_rate,  color='y',  linestyle='--',  alpha=0.7,
label='Baseline')            self.ax1.set_title('Heart Rate', color='#ffffff', fontsize=12)
self.ax1.set_xlabel('Time (samples)', color='#ffffff')        self.ax1.set_ylabel('BPM',
color='#ffffff')          self.ax1.tick_params(colors='#ffffff')          self.ax1.grid(True,
alpha=0.3)        self.ax1.legend()

        # Plot stress levels        self.ax2.plot(time_indices, list(self.stress_levels),
'r-', linewidth=2)
        self.ax2.axhline(y=self.stress_thresholds['low'], color='g', linestyle='--', alpha=0.7,
label='Low')
```

```
        self.ax2.axhline(y=self.stress_thresholds['moderate'], color='y', linestyle='--', alpha=0.7,
label='Moderate')          self.ax2.axhline(y=self.stress_thresholds['high'], color='r',
linestyle='--', alpha=0.7, label='High')          self.ax2.set_title('Stress Level',
color='#ffffff', fontsize=12)          self.ax2.set_xlabel('Time (samples)', color='#ffffff')
self.ax2.set_ylabel('Stress', color='#ffffff')          self.ax2.tick_params(colors='#ffffff')
self.ax2.grid(True, alpha=0.3)          self.ax2.legend()

        self.canvas.draw()


    def toggle_monitoring(self):
        self.is_monitoring = not self.is_monitoring


        if self.is_monitoring:                # Start monitoring session                self.session_start_time =
datetime.datetime.now()             self.session_data = []             self.start_button.config(text="■
Stop", bg="#ff4444")             self.status_label.config(text="Monitoring", fg="#00ff41")
self.monitoring_thread = threading.Thread(target=self.monitoring_loop, daemon=True)
self.monitoring_thread.start()          else:
            # Stop monitoring and save session             self.session_end_time
= datetime.datetime.now()             self.start_button.config(text="▶
Start", bg="#00ff41")             self.status_label.config(text="Stopped",
fg="#ffffff")


            # Save session data to database             self.save_session_to_database()


            # Stop device monitoring if active             if hasattr(self,
'device_connected') and self.device_connected:
                try:
                    device_connector.stop_monitoring()
```

```
            except:
pass


        # Prompt user to save/export
save_prompt = messagebox.askyesno(
            "Session Ended",
            "Monitoring session has ended.\n\nWould you like to save this session to the database and export to PDF?"
        )
        if save_prompt:
            self.export_session_to_pdf()


    def monitoring_loop(self):
while self.is_monitoring:
        if self.is_monitoring:  # Double check before updating              self.root.after(0, self.update_display)            time.sleep(1)  # Update every second


    def calibrate_baseline(self):
if not self.is_monitoring:
        messagebox.showinfo("Calibration", "Please start monitoring first to calibrate baseline.")
        return


      # Calculate average heart rate from recent data        if len(self.heart_rate_data) >= 10:
        recent_hr = list(self.heart_rate_data)[-10:]           self.baseline_heart_rate = sum(recent_hr) / len(recent_hr)
```

```
            messagebox.showinfo("Calibration", f"Baseline calibrated to
{int(self.baseline_heart_rate)} BPM")
        else:
            messagebox.showwarning("Calibration", "Need more data to calibrate. Please monitor for
at least 10 seconds.")


    def trigger_alert(self):
        # Only trigger alert if cooldown period has passed
current_time = time.time()          if current_time - self.last_alert_time
>= self.alert_cooldown:
            self.last_alert_time = current_time
# Simple alert notification              self.root.bell()  # System beep              alert_msg = f"High
stress detected! Current level: {self.current_stress_level:.2f}\n\nNext
alert in {self.alert_cooldown // 60:.0f} minutes."              messagebox.showwarning("Stress Alert",
alert_msg)


    def store_data_point(self):
        data_point = {
            'timestamp': datetime.datetime.now().isoformat(),
            'heart_rate': self.current_heart_rate,
            'stress_level': self.current_stress_level
        }
        self.historical_data.append(data_point)


        # Keep only last 1000 data points          if
len(self.historical_data) > 1000:
            self.historical_data = self.historical_data[-1000:]
```

```
    def show_history(self):
        if not self.historical_data:
            messagebox.showinfo("History", "No historical data available. Start monitoring to collect data.")
return

        # Create history window        history_window
= tk.Toplevel(self.root)
history_window.title("Historical Data")
history_window.geometry("1000x700")
history_window.configure(bg='#1e1e1e')

        # Title              title_label = tk.Label(history_window, text="Historical Data
Analysis",                        font=('Arial', 16, 'bold'), fg='#00ff41', bg='#1e1e1e')
title_label.pack(pady=10)

        # Statistics frame                    stats_frame = tk.Frame(history_window, bg='#2d2d2d',
relief=tk.RAISED, bd=2)        stats_frame.pack(fill=tk.X, padx=20, pady=(0, 10))

        # Calculate statistics        if
self.historical_data:
            hr_values = [data['heart_rate'] for data in self.historical_data]            stress_values
= [data['stress_level'] for data in self.historical_data]

            avg_hr = sum(hr_values) / len(hr_values)            max_hr
= max(hr_values)
            min_hr = min(hr_values)            avg_stress = sum(stress_values)
/ len(stress_values)            max_stress
```

```
= max(stress_values)            min_stress = min(stress_values)


        # Display statistics            stats_text
= f"""
Total Records: {len(self.historical_data)} |
Avg HR: {avg_hr:.1f} BPM |
HR Range: {min_hr:.1f} - {max_hr:.1f} BPM |
Avg Stress: {avg_stress:.3f} |
Stress Range: {min_stress:.3f} - {max_stress:.3f}
        """


        tk.Label(stats_frame, text=stats_text, font=('Arial', 10),                  fg='#ffffff',
bg='#2d2d2d', justify=tk.LEFT).pack(padx=10, pady=5)

    # Control frame for filtering        control_frame =
tk.Frame(history_window, bg='#1e1e1e')        control_frame.pack(fill=tk.X,
padx=20, pady=(0, 10))


    # Filter controls
    tk.Label(control_frame, text="Filter:", fg='#ffffff', bg='#1e1e1e').pack(side=tk.LEFT,
padx=(0, 5))


    # Date range filter
    tk.Label(control_frame, text="From:", fg='#ffffff', bg='#1e1e1e').pack(side=tk.LEFT,
padx=(10,   2))     from_date_var   =
    tk.StringVar()
    from_date_entry = tk.Entry(control_frame, textvariable=from_date_var, bg='#2d2d2d',
```

```
fg='#ffffff', width=12)        from_date_entry.pack(side=tk.LEFT,
padx=(0, 5))
      from_date_entry.insert(0, (datetime.datetime.now() -
datetime.timedelta(days=7)).strftime('%Y-%m-%d'))

      tk.Label(control_frame, text="To:", fg='#ffffff', bg='#1e1e1e').pack(side=tk.LEFT, padx=(5,
2))        to_date_var = tk.StringVar()         to_date_entry = tk.Entry(control_frame,
textvariable=to_date_var, bg='#2d2d2d', fg='#ffffff', width=12)
to_date_entry.pack(side=tk.LEFT, padx=(0, 10))        to_date_entry.insert(0,
datetime.datetime.now().strftime('%Y-%m-%d'))

      # Stress level filter         tk.Label(control_frame, text="Stress Level:", fg='#ffffff',
bg='#1e1e1e').pack(side=tk.LEFT, padx=(10, 2))        stress_filter_var =
tk.StringVar(value="All")        stress_filter = ttk.Combobox(control_frame,
textvariable=stress_filter_var,                                values=["All", "Low", "Moderate", "High",
"Critical"],                              width=10, state='readonly')
stress_filter.pack(side=tk.LEFT, padx=(0, 10))

      # Refresh button        def
refresh_history():            # Clear existing
data            for item in
tree.get_children():
            tree.delete(item)

         # Apply filters            filtered_data = self.historical_data.copy()

         # Date filter            try:
            from_date = datetime.datetime.strptime(from_date_var.get(), '%Y-%m-%d')
```

```
            to_date = datetime.datetime.strptime(to_date_var.get(), '%Y-%m-%d') +
datetime.timedelta(days=1)                  filtered_data =
[data for data in filtered_data
                        if from_date <= datetime.datetime.fromisoformat(data['timestamp']) < to_date]
        except:
pass


        # Stress level filter
stress_filter_value = stress_filter_var.get()           if
stress_filter_value != "All":
            filtered_data = [data for data in filtered_data                                  if
self.get_stress_category(data['stress_level'])[0] == stress_filter_value]


        # Add filtered data           for data in filtered_data[-
100:]:  # Show last 100 entries
            timestamp = datetime.datetime.fromisoformat(data['timestamp']).strftime('%Y-%m-%d
%H:%M:%S')
            stress_category, _ = self.get_stress_category(data['stress_level'])              tree.insert(",
tk.END, values=(timestamp, f"{data['heart_rate']:.1f}",
f"{data['stress_level']:.3f}", stress_category)) tk.Button(control_frame, text="Refresh",
command=refresh_history,                bg='#00ff41', fg='#1e1e1e', font=('Arial', 10,
'bold')).pack(side=tk.LEFT, padx=(0, 10))


    # Export button for history
def export_history():           if not
tree.get_children():
            messagebox.showinfo("Export", "No data to export.")
```

```
            return

        filename =
f"history_export_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
        try:                with
open(filename, 'w') as f:
                f.write("Timestamp,Heart Rate,Stress Level,Category\n")                for
child in tree.get_children():
                    values = tree.item(child)['values']
                    f.write(f"{values[0]},{values[1]},{values[2]},{values[3]}\n")
messagebox.showinfo("Export Success", f"History exported to {filename}")            except
Exception as e:
            messagebox.showerror("Export Error", f"Failed to export: {str(e)}")

    tk.Button(control_frame,          text="Export          CSV",          command=export_history,
bg='#0080ff', fg='#ffffff', font=('Arial', 10, 'bold')).pack(side=tk.LEFT)

    # Create treeview for data display with scrollbars        tree_frame =
tk.Frame(history_window, bg='#2d2d2d', relief=tk.RAISED, bd=2)
tree_frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=(0, 10))

    # Scrollbars        tree_scroll_y = tk.Scrollbar(tree_frame,
orient=tk.VERTICAL)        tree_scroll_x = tk.Scrollbar(tree_frame,
orient=tk.HORIZONTAL)
```

```
# Treeview columns = ('Timestamp', 'Heart Rate', 'Stress Level', 'Category') tree =
ttk.Treeview(tree_frame, columns=columns, show='headings', height=20,
yscrollcommand=tree_scroll_y.set, xscrollcommand=tree_scroll_x.set)

# Configure scrollbars tree_scroll_y.config(command=tree.yview)
tree_scroll_x.config(command=tree.xview)

# Configure columns
tree.heading('Timestamp', text='Timestamp', command=lambda: self.sort_column(tree,
'Timestamp', False))
tree.heading('Heart Rate', text='Heart Rate (BPM)', command=lambda: self.sort_column(tree,
'Heart Rate', False))
tree.heading('Stress Level', text='Stress Level', command=lambda: self.sort_column(tree,
'Stress Level', False))
tree.heading('Category', text='Category', command=lambda: self.sort_column(tree,
'Category', False))

tree.column('Timestamp', width=180) tree.column('Heart
Rate', width=120) tree.column('Stress
Level', width=100) tree.column('Category', width=100)

# Grid layout for treeview and scrollbars
tree.grid(row=0, column=0, sticky='nsew')
tree_scroll_y.grid(row=0, column=1, sticky='ns')
tree_scroll_x.grid(row=1, column=0, sticky='ew')

tree_frame.grid_rowconfigure(0, weight=1) tree_frame.grid_columnconfigure(0,
weight=1)
```

```
        # Add initial data         refresh_history()

        #     Close     button                                  tk.Button(history_window,     text="Close",
command=history_window.destroy,
                 bg='#ff4444', fg='#ffffff', font=('Arial', 10, 'bold'),
padx=20, pady=5).pack(pady=10)

    def sort_column(self, tree, col, reverse):          """Sort treeview
column when clicked"""          data = [(tree.set(item, col), item) for item
in tree.get_children('')]

        # Try to sort numerically for Heart Rate and Stress Level          if
col in ['Heart Rate', 'Stress Level']:
            try:
                data.sort(key=lambda x: float(x[0]), reverse=reverse)             except:

                data.sort(reverse=reverse)
else:
data.sort(reverse=reverse)

        for index, (val, item) in enumerate(data):
            tree.move(item, '', index)

        tree.heading(col, command=lambda: self.sort_column(tree, col, not reverse))

    def export_data(self):          if
not self.historical_data:
```

```
            messagebox.showinfo("Export", "No data to export.")
            return

        # Create export menu          export_window
= tk.Toplevel(self.root)
export_window.title("Export Options")
export_window.geometry("400x200")
export_window.configure(bg='#1e1e1e')

        tk.Label(export_window,    text="Select    Export    Format",    font=('Arial',    14,    'bold'),
fg='#00ff41', bg='#1e1e1e').pack(pady=10)

        # JSON export          tk.Button(export_window, text="
Export as JSON",                  command=lambda:
self.export_json(),                  bg='#0080ff', fg='#ffffff', font=('Arial',
12, 'bold'),                padx=20, pady=10, width=30).pack(pady=5)

        # PDF export          tk.Button(export_window, text="
Export as PDF",                  command=lambda:
self.export_session_to_pdf(),                  bg='#ff1493', fg='#ffffff',
font=('Arial', 12, 'bold'),                  padx=20,
pady=10, width=30).pack(pady=5)

        # Both exports          tk.Button(export_window, text="  Export Both",
command=lambda: [self.export_json(), self.export_session_to_pdf()],
bg='#32cd32', fg='#000000', font=('Arial', 12, 'bold'),                  padx=20, pady=10,
width=30).pack(pady=5)
```

```
    def export_json(self):
        """Export historical data to JSON format"""
        if not self.historical_data:
            messagebox.showinfo("Export", "No data to export.")
            return

        filename = f"stress_data_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.json"
        try:
            with open(filename, 'w') as f:
                json.dump(self.historical_data, f, indent=2)
            messagebox.showinfo("Export", f"Data exported to {filename}")
        except Exception as e:
            messagebox.showerror("Export Error", f"Failed to export data: {str(e)}")

    def export_session_to_pdf(self):
        """Export current or recent monitoring session to PDF"""
        if not self.historical_data and not self.session_data:
            messagebox.showinfo("Export", "No session data to export.")
            return

        # Use file dialog to save PDF
        file_path = filedialog.asksaveasfilename(
            defaultextension=".pdf",
            filetypes=[("PDF files", "*.pdf"), ("All files", "*.*")],
            initialfile=f"stress_report_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.pdf"
```

```
        )


        if not file_path:
            return
                try:
            self._create_pdf_report(file_path)
            messagebox.showinfo("Success", f"PDF report saved to:\n{file_path}\n\nFile is ready for
download!")
        except Exception as e:
            messagebox.showerror("PDF Export Error", f"Failed to create PDF: {str(e)}")


    def _create_pdf_report(self, filename):
        """Create a detailed PDF report of monitoring data"""
        # Create PDF document         doc =
SimpleDocTemplate(filename, pagesize=letter)
        story = []         styles =
getSampleStyleSheet()


        # Custom styles         title_style
= ParagraphStyle(
            'CustomTitle',             parent=styles['Heading1'],
fontSize=24,
textColor=colors.HexColor('#00ff41'),
spaceAfter=30,
alignment=TA_CENTER
        )
```

```
        heading_style = ParagraphStyle(
'CustomHeading',
parent=styles['Heading2'],
fontSize=14,
textColor=colors.HexColor('#00ff41'),
spaceAfter=12,            spaceBefore=12
        )

        # Title        story.append(Paragraph("Smart Wearable Stress Monitor - Report",
title_style))        story.append(Spacer(1, 0.3*inch))


        # Session info        if
self.session_start_time:
session_info = f"""
            <b>Session Information:</b><br/>
            Started: {self.session_start_time.strftime('%Y-%m-%d %H:%M:%S')}<br/>
            """            if
self.session_end_time:
                session_info += f"Ended: {self.session_end_time.strftime('%Y-%m-%d
%H:%M:%S')}<br/>"
                duration = (self.session_end_time - self.session_start_time).total_seconds()
minutes = int(duration // 60)                seconds = int(duration % 60)
session_info += f"Duration: {minutes}m {seconds}s<br/>"
story.append(Paragraph(session_info, styles['Normal']))            story.append(Spacer(1,
0.2*inch))


        # Statistics        if self.historical_data:
```

```
        hr_values = [data['heart_rate'] for data in self.historical_data]          stress_values
= [data['stress_level'] for data in self.historical_data]

        avg_hr = sum(hr_values) / len(hr_values)          max_hr
= max(hr_values)          min_hr = min(hr_values)
avg_stress = sum(stress_values) / len(stress_values)
max_stress = max(stress_values)


        stats_data = [
            ['Metric', 'Value'],
            ['Total Records', str(len(self.historical_data))],
            ['Average Heart Rate', f'{avg_hr:.1f} BPM'],
            ['Max Heart Rate', f'{max_hr:.1f} BPM'],
            ['Min Heart Rate', f'{min_hr:.1f} BPM'],
            ['Average Stress Level', f'{avg_stress:.3f}'],
            ['Max Stress Level', f'{max_stress:.3f}'],
            ['Baseline Heart Rate', f'{self.baseline_heart_rate:.1f} BPM'],
        ]

        stats_table = Table(stats_data, colWidths=[3*inch, 2*inch])
stats_table.setStyle(TableStyle([
            ('BACKGROUND', (0, 0), (-1, 0), colors.HexColor('#00ff41')),
            ('TEXTCOLOR', (0, 0), (-1, 0), colors.black),
            ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
            ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
            ('FONTSIZE', (0, 0), (-1, 0), 12),
            ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
```

```
            ('BACKGROUND', (0, 1), (-1, -1), colors.HexColor('#f0f0f0')),
            ('GRID', (0, 0), (-1, -1), 1, colors.black),
        ]))

        story.append(Paragraph("<b>Session Statistics:</b>", heading_style))
story.append(stats_table)        story.append(Spacer(1, 0.3*inch))


        # Sample data table
story.append(PageBreak())
        story.append(Paragraph("<b>Detailed Readings (Latest 20 Records):</b>",
heading_style))

        recent_data = self.historical_data[-20:]        data_table_data
= [['Time', 'Heart Rate (BPM)', 'Stress Level']]

        for data in recent_data:
            timestamp = datetime.datetime.fromisoformat(data['timestamp'])
data_table_data.append([              timestamp.strftime('%H:%M:%S'),
f"{data['heart_rate']:.1f}",              f"{data['stress_level']:.3f}"
            ])

        data_table = Table(data_table_data, colWidths=[1.5*inch, 2*inch, 2*inch])
data_table.setStyle(TableStyle([
            ('BACKGROUND', (0, 0), (-1, 0), colors.HexColor('#00ff41')),
            ('TEXTCOLOR', (0, 0), (-1, 0), colors.black),
            ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
            ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
```

```
            ('FONTSIZE', (0, 0), (-1, 0), 10),
            ('BOTTOMPADDING', (0, 0), (-1, 0), 8),
            ('BACKGROUND', (0, 1), (-1, -1), colors.HexColor('#f0f0f0')),
            ('GRID', (0, 0), (-1, -1), 1, colors.black),
            ('FONTSIZE', (0, 1), (-1, -1), 8),
        ]))

        story.append(data_table)

    # Footer    story.append(Spacer(1, 0.3*inch))    story.append(Paragraph(    f"Report
Generated: {datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')}",
        styles['Normal']
    ))

    # Build PDF    doc.build(story)

  def init_database(self):
    """Initialize SQLite database for storing monitoring sessions"""    self.db_path
= 'stress_monitor.db'
    try:
        conn = sqlite3.connect(self.db_path)        cursor
= conn.cursor()

        # Create sessions table        cursor.execute('''
            CREATE TABLE IF NOT EXISTS sessions (
id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
                user_id TEXT,
session_name TEXT,
start_time TIMESTAMP,
end_time TIMESTAMP,
duration_seconds INTEGER,
avg_heart_rate REAL,
max_heart_rate REAL,
min_heart_rate REAL,
avg_stress REAL,
max_stress REAL,
record_count
INTEGER,              json_data TEXT,              created_at TIMESTAMP
DEFAULT CURRENT_TIMESTAMP
            )
        ''')

        # Create readings table          cursor.execute('''
            CREATE TABLE IF NOT EXISTS readings (
id INTEGER PRIMARY KEY AUTOINCREMENT,
                session_id      INTEGER,
timestamp       TIMESTAMP,                      heart_rate
REAL,
stress_level REAL,
                FOREIGN KEY(session_id) REFERENCES sessions(id)
            )
        ''')
```

```
        conn.commit()           conn.close()
print("Database initialized successfully")        except
Exception as e:
        print(f"Database initialization error: {e}")


    def save_session_to_database(self):
        """Save current monitoring session to database"""        if
not self.historical_data or not self.session_start_time:
        return
            try:
        conn = sqlite3.connect(self.db_path)            cursor
= conn.cursor()

        # Calculate statistics            hr_values = [data['heart_rate'] for data
in self.historical_data]            stress_values = [data['stress_level'] for data
in self.historical_data]

        avg_hr = sum(hr_values) / len(hr_values) if hr_values else 0
max_hr = max(hr_values) if hr_values else 0            min_hr = min(hr_values) if
hr_values else 0            avg_stress = sum(stress_values) / len(stress_values) if
stress_values else 0            max_stress = max(stress_values) if stress_values else 0

        duration = (self.session_end_time - self.session_start_time).total_seconds() if
self.session_end_time else 0

        # Insert session
        session_name = f"Session {self.session_start_time.strftime('%Y-%m-%d %H:%M')}"
```

```
        cursor.execute('''
            INSERT INTO sessions (user_id, session_name, start_time, end_time,
duration_seconds,
                                  avg_heart_rate, max_heart_rate, min_heart_rate,
avg_stress, max_stress, record_count, json_data)                VALUES (?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?)
        ''', (
            get_current_user() or 'unknown',
session_name,                self.session_start_time.isoformat(),
            (self.session_end_time.isoformat() if self.session_end_time else None),
            int(duration),                avg_hr,
max_hr, min_hr,                avg_stress,
max_stress,                len(self.historical_data),
json.dumps(self.historical_data)
        ))

        session_id = cursor.lastrowid

        # Insert readings            for data
in self.historical_data:
cursor.execute('''
                INSERT INTO readings (session_id, timestamp, heart_rate, stress_level)
VALUES (?, ?, ?, ?)
            ''', (                    session_id,
data['timestamp'],
data['heart_rate'],
data['stress_level']
            ))
```

```
        conn.commit()        conn.close()        print(f"Session
saved to database with ID: {session_id}")      except Exception
as e:
        print(f"Error saving session to database: {e}")


   def load_historical_data(self):
# Load previous data if exists
      try:
        # This would load from a file in a real implementation
        pass      except:
pass


   def setup_sensors(self):
      """Setup available sensors"""                    # Add simulated sensor
self.sensor_manager.add_sensor(create_sensor("simulated", base_heart_rate=72))
# Add Bluetooth sensor placeholder

      self.sensor_manager.add_sensor(create_sensor("bluetooth",
device_address="00:11:22:33:44:55"))


      # Add serial sensor placeholder      self.sensor_manager.add_sensor(create_sensor("serial",
port="COM3", baudrate=9600))
```

APPENDIX E

SYSTEM RESULTS

Section A: Dashboard

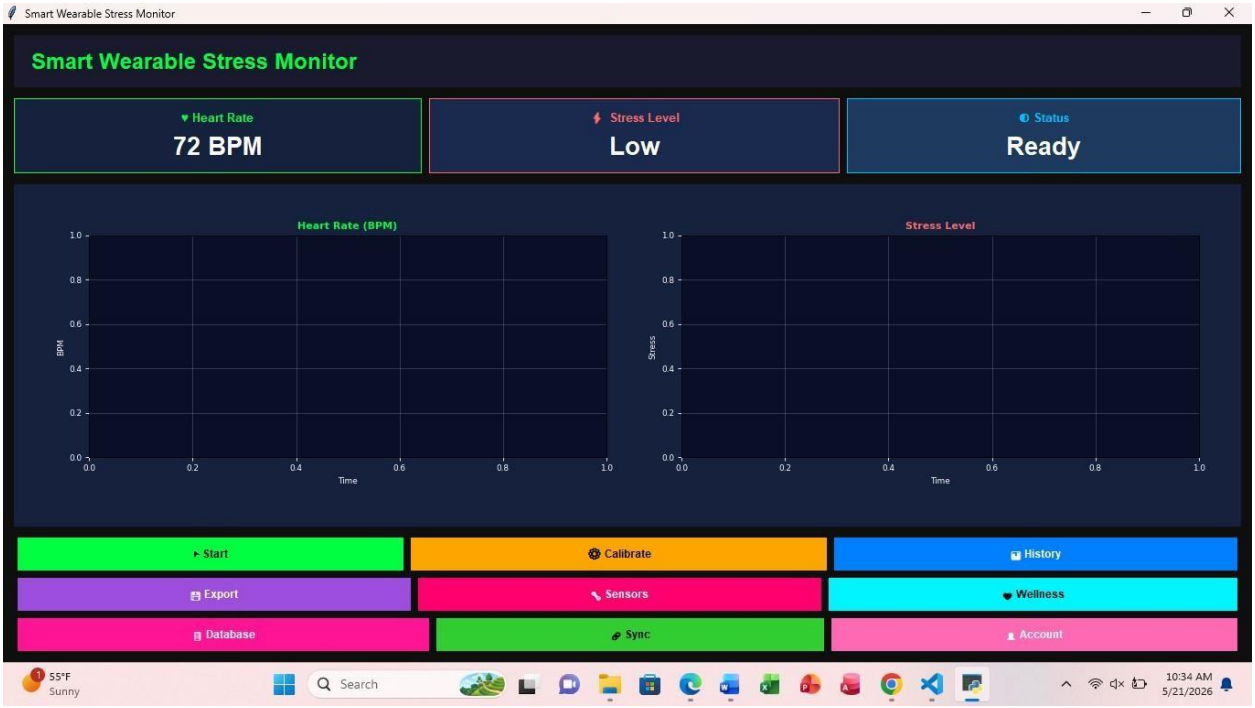


Figure 20: E1 Dashboard

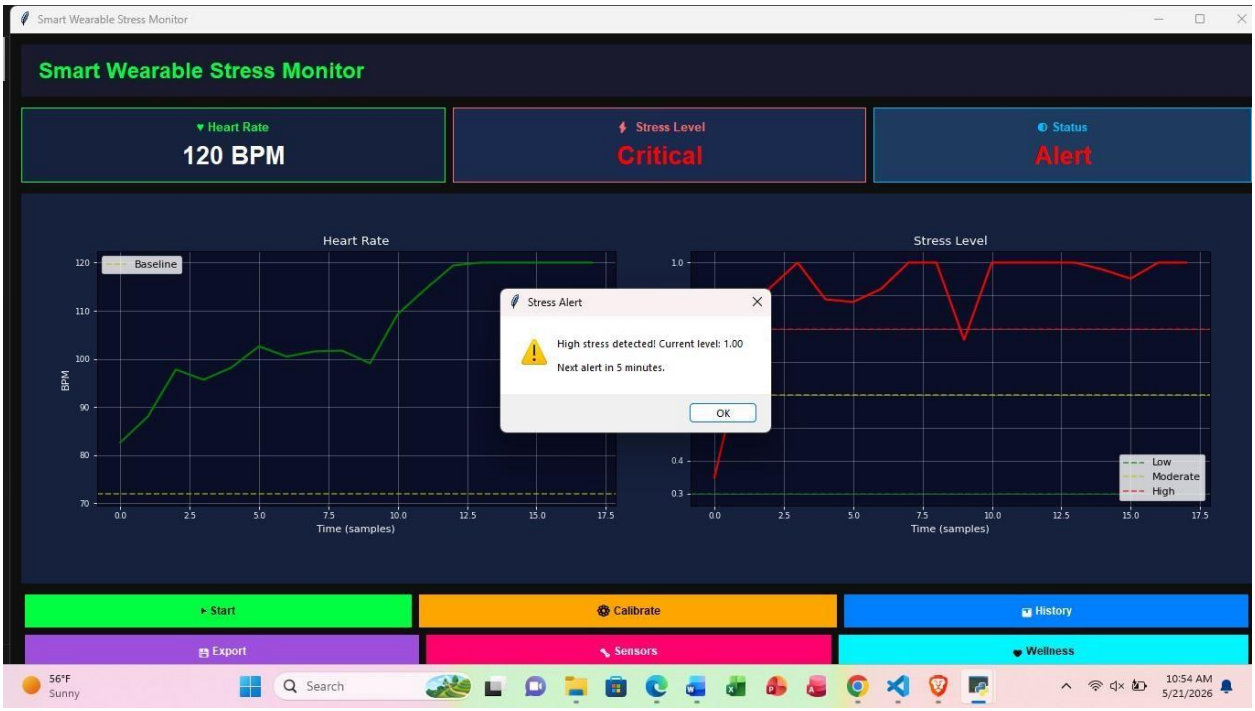


Figure 21: E2 Stress Alerts

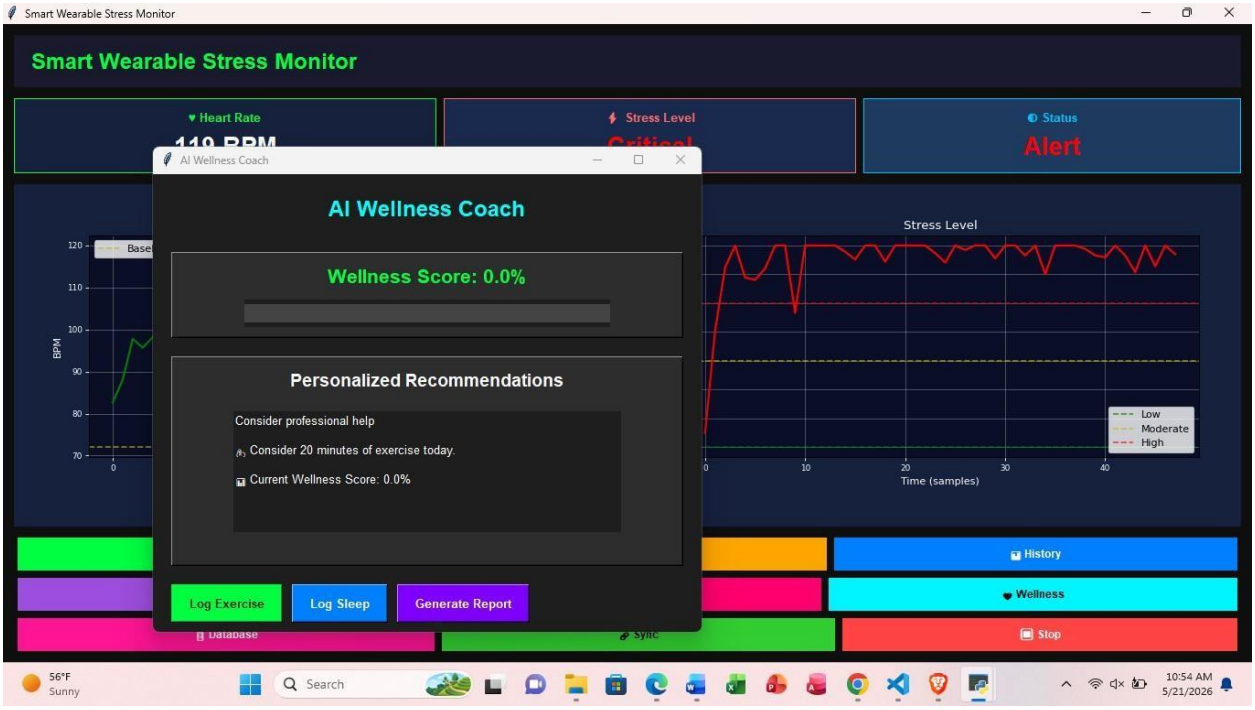


Figure 22: E3 Wellness Mentor

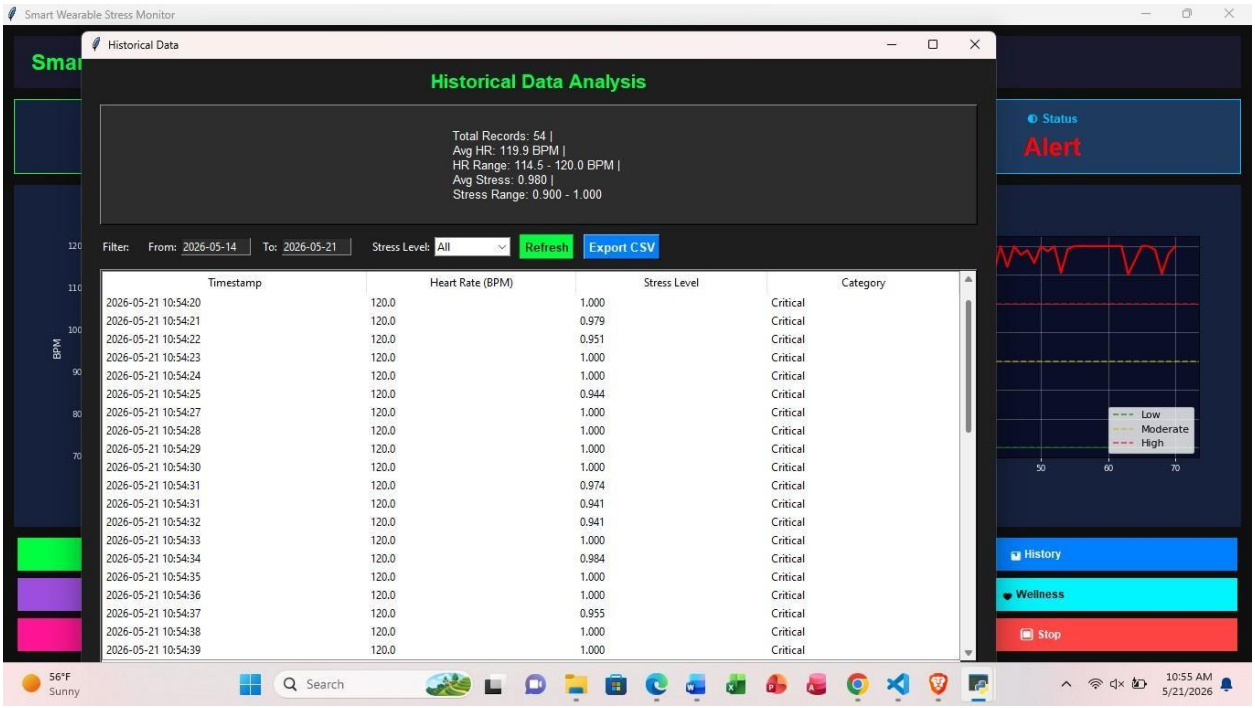


Figure 23: E4 Historical Data Analysis

REFERENCES

1. Bassi, A. and Horn, G. (2020) *The Internet of Things: Opportunities for health improvement in developing nations*. Health Informatics International, 11(2), pp. 88–97.
2. Gjoreski, M., Gjoreski, H., Luštrek, M. and Gams, M. (2020) 'Continuous stress detection using wearable sensors: A review', *IEEE Transactions on Affective Computing*, 11(2), pp. 394–406.
3. Karthikeyan, P. and Somasundaram, S. (2021) 'Non-invasive wearable systems for stress detection using machine learning', *Biomedical Signal Processing and Control*, 70, p. 103041.
4. Kim, J. and Kim, K. (2022) 'A smartwatch-based sleep monitoring system using HRV and motion sensors', *Sensors*, 22(9), pp. 3578–3590.
5. Moyo, C. and Chikasha, T. (2023) 'IoT applications in health monitoring in Zimbabwe', *Zimbabwe Journal of Applied Science & Technology*, 8(1), pp. 101–113.
6. Wheaton, B., 1997. The nature of chronic stress. In *Coping with chronic stress* (pp. 43-73). Boston, MA: Springer US.
7. World Health Organization (2023) *Sleep and Mental Health: Global Overview*. Geneva: WHO Press.

8. Deng, J., Huang, S., Wang, L., Deng, W. and Yang, T., 2022. Conceptual framework for smart health: A multi-dimensional model using IPO logic to link drivers and outcomes. International Journal of Environmental Research and Public Health, 19(24), p.16742.
9. Dannels, S.A., 2018. Research design. In *The reviewer's guide to quantitative methods in the social sciences* (pp. 402-416). Routledge.
10. Leech, N.L., Dellinger, A.B., Brannagan, K.B. and Tanaka, H., 2010. Evaluating mixed research studies: A mixed methods approach. *Journal of mixed methods research*, *4*(1), pp.17-31.
11. Barreiro, P.L. and Albandoz, J.P., 2001. Population and sample. Sampling techniques. *Management mathematics for European schools*, *1*(1), pp.1-18.
12. Hinds, D., 2002. Research instruments. In *The researcher's toolkit* (pp. 57-58). Routledge.