**DevRelax: STRESS MONITORING AND RELIEVING APPLICATION FOR IT PROFESSIONALS**

Perera M. S. D

B.Sc. (Hons) Degree in Information Technology

(Specialization in Software Engineering)

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

**DevRelax: STRESS MONITORING AND RELIEVING APPLICATION FOR IT PROFESSIONALS**

Perera M. S. D IT20020262

The dissertation was submitted in partial fulfilment of the requirements

for the B.Sc. Special Honors degree in Information Technology

(Specialization in Software Engineering)

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

# DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Signature |
| Perera M. S. D | IT20020262 |  |

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor Date

 10th September 2023

(Mr. Samadhi Rathnayake)

# ABSTRACT

The proposed solution discussed in this report focuses on detecting stress levels of IT professionals. This utilizes a machine learning approach and comprises four key components. The first two components analyze keystroke dynamics and heart rate variability, respectively, providing crucial inputs for stress level assessment. The third component employs facial dynamics analysis to detect emotional states, a key indicator of stress levels. The final component offers personalized recommendations and suggestions to help users alleviate detected stress levels.

tress is a pervasive concern among IT professionals, and its long-term effects on physical and mental health are well-documented. Early detection and intervention are imperative. This system not only identifies stress levels promptly but also provides tailored strategies to mitigate them. By doing so, it aims to improve the overall work experience, reduce health complications, and enhance productivity for IT professionals.

Furthermore, this report emphasizes more on the implementation, functionality, requirements, and other aspects of the keystroke dynamic based stress detection component.

Keywords: Machine learning, Random Forest Model, IT Professionals, Keystroke Dynamics, Python Program, Stress Detection, Aggregation

# ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my project supervisor, Mr. Samadhi Rathnayake, for his unwavering guidance and invaluable support throughout my undergraduate research. His expertise and mentorship played a huge role in the successful completion of this project. I am also immensely thankful to Mrs. Devanshi Ganegoda, my co-supervisor, for her continuous assistance and insightful contributions, which significantly enriched the quality of this research endeavor. Furthermore, I extend my heartfelt appreciation to Mrs. Arosha Dassanayake, our external supervisor, whose expertise and dedicated involvement elevated the accuracy and precision of our research, despite her demanding schedule. Also, I would like to extend my gratitude to our lecturer in charge, Dr. Jayantha Amararachchi, and the CDAP team for their invaluable assistance in ensuring the success of this project. I am also deeply appreciative of my friends and teammates for their unwavering encouragement and support throughout this endeavor. Their contributions have been instrumental. Lastly, I want to express my heartfelt thanks to my parents for their enduring support and encouragement. Their belief in me has been a constant source of inspiration.

# TABLE OF CONTENTS

[DECLARATION i](#_Toc145304831)

[ABSTRACT ii](#_Toc145304832)

[ACKNOWLEDGEMENT iii](#_Toc145304833)

[TABLE OF CONTENTS iv](#_Toc145304834)

[LIST OF FIGURES vi](#_Toc145304835)

[LIST OF TABLES ix](#_Toc145304836)

[LIST OF ABBREVIATIONS x](#_Toc145304837)

[1. INTRODUCTION 1](#_Toc145304838)

[1.1 Background and Literature 2](#_Toc145304839)

[1.2 Research Gap 4](#_Toc145304840)

[1.3 Research Problem 7](#_Toc145304841)

[2. RESEARCH OBJECTIVES 9](#_Toc145304842)

[2.1 Main Objectives 9](#_Toc145304843)

[2.2 Specific Objectives 10](#_Toc145304844)

[3. METHODOLOGY 11](#_Toc145304845)

[3.1. Methodology 11](#_Toc145304846)

[3.1.1. Data Collection 11](#_Toc145304847)

[3.1.2. Approach 12](#_Toc145304848)

[3.1.3. Tools and Technologies 15](#_Toc145304849)

[3.2. Commercialization Aspect of the Product 16](#_Toc145304850)

[3.3. Testing and Implementation 18](#_Toc145304851)

[3.3.1. Implementation 18](#_Toc145304852)

[3.3.2. Functional and Non-Functional Requirements 43](#_Toc145304853)

[3.3.3. Feasibility Study 44](#_Toc145304854)

[3.3.4. Testing 45](#_Toc145304855)

[4. RESULTS AND DISCUSSION 47](#_Toc145304856)

[4.1. Results 47](#_Toc145304857)

[4.2. Research Findings 59](#_Toc145304858)

[4.3. Discussion 60](#_Toc145304859)

[4.4. Limitations 62](#_Toc145304860)

[4.5. Summary of Each Students Contribution 63](#_Toc145304861)

[5. CONCLUSION 65](#_Toc145304862)

[6. REFERENCES 67](#_Toc145304863)

[7. APPENDICES 69](#_Toc145304864)

# LIST OF FIGURES

[Figure 3. 1: Keystroke dynamic based stress detection component diagram 12](file:///D:/SLIIT/4TH%20YEAR/1st%20Semester/Research%20Project/Final%20report/IT20020262_Final_Report_Draft.docx#_Toc145280782)

[Figure 3. 2: Commercialization Discussion 16](#_Toc145280783)

[Figure 3. 3: Dataset Before Preprocessing 17](#_Toc145280784)

[Figure 3. 4: Dataset 2 Before Preprocessing 18](#_Toc145280785)

[Figure 3. 5: Label Encoding on Dataset 19](#_Toc145280786)

[Figure 3. 6: Extracting Hour and Day of Week data 19](#_Toc145280787)

[Figure 3. 7: One Hot Encoding on the dataset 20](#_Toc145280788)

[Figure 3. 8: RNN LSTM Train Test Split 20](#_Toc145280789)

[Figure 3. 9: ARIMA Train and Test Set 21](#_Toc145280790)

[Figure 3. 10: SVM Model Train and Test Set 21](#_Toc145280791)

[Figure 3. 11: Random Forest Train and Test Split 22](#_Toc145280792)

[Figure 3. 12: Before Oversampling 22](#_Toc145280793)

[Figure 3. 13: Oversampling 23](#_Toc145280794)

[Figure 3. 14: After Oversampling 23](#_Toc145280795)

[Figure 3. 15: Finalized Dataset 24](#_Toc145280796)

[Figure 3. 16: Keylogger Functionality 25](#_Toc145280797)

[Figure 3. 17: Gathered keystroke dynamics snapshot 26](#_Toc145280798)

[Figure 3. 18: Keylogger Functions 28](#_Toc145280799)

[Figure 3. 19: Generating dump file of the ML model using pickle library 28](#_Toc145280800)

[Figure 3. 20: Session timer 29](#_Toc145280801)

[Figure 3. 21: Send Generated CSV file to the ML model code snippet 29](#_Toc145280802)

[Figure 3. 22: Loading model dump from server 29](#_Toc145280803)

[Figure 3. 23: Prediction function 30](#_Toc145280804)

[Figure 3. 24: Prediction result fetch function 30](#_Toc145280805)

[Figure 3. 25: Stress Level Handler code snippet for handling combinations 31](#_Toc145280806)

[Figure 3. 26: Stress Level Handler code snippet for final output 32](#_Toc145280807)

[Figure 3. 27: Retrieve Aggregated Stress Value 32](#_Toc145280808)

[Figure 3. 28: Stress level appending code snippet (continuation of update\_csv function) 33](#_Toc145280809)

[Figure 3. 29: After appending the Aggregated Stress Level 33](#_Toc145280810)

[Figure 3. 30: Append the file to the combined dataset file 34](#_Toc145280811)

[Figure 3. 31: CSV file after it has been appended to the combined dataset 34](#_Toc145280812)

[Figure 3. 32: Incremental Model Handler loading model and testing current accuracy code snippet 35](#_Toc145280813)

[Figure 3. 33: Model Retraining and Testing New Accuracy 36](#_Toc145280814)

[Figure 3. 34: Compare old and new model accuracy 36](#_Toc145280815)

[Figure 3. 35: Scheduler Function 37](#_Toc145280816)

[Figure 3. 36: Calling the function in 24 hours 37](#_Toc145280817)

[Figure 3. 37: Running the job in a new thread 37](#_Toc145280818)

[Figure 3. 38: Frontend APIs 38](#_Toc145280819)

[Figure 3. 39: Displaying gathered prediction data 38](#_Toc145280820)

[Figure 3. 40: Keystroke Dynamic based stress data retrieve function snippet 39](#_Toc145280821)

[Figure 3. 41: OnDemand keystroke-based stress retrieve function 39](#_Toc145280822)

[Figure 3. 42: Typing test dashboard 40](#_Toc145280823)

[Figure 3. 43: Retrain handler code snippet 40](#_Toc145280824)

[Figure 4. 1: Predict Test Result 46](#_Toc145280825)

[Figure 4. 2: Ondemand Predict Test Result 46](#_Toc145280826)

[Figure 4. 3: Ondemand Test Result 47](#_Toc145280827)

[Figure 4. 4: get\_data Test Result 47](#_Toc145280828)

[Figure 4. 5: update\_csv Test Result 48](#_Toc145280829)

[Figure 4. 6: Accuracy of the model without oversampling (DS-1) 49](#_Toc145280830)

[Figure 4. 7: Confusion Matrix Heatmap of the model without oversampling (DS-1) 50](#_Toc145280831)

[Figure 4. 8: Accuracy of the model after oversampling (DS-1) 50](#_Toc145280832)

[Figure 4. 9: Confusion Matrix Heatmap of the model after oversampling (DS-1) 51](#_Toc145280833)

[Figure 4. 10: Accuracy of the model without oversampling (DS-2) 51](#_Toc145280834)

[Figure 4. 11: Confusion Matrix Heatmap of the model without oversampling (DS-2) 52](#_Toc145280835)

[Figure 4. 12: Accuracy of the model after oversampling (DS-2) 52](#_Toc145280836)

[Figure 4. 13: Confusion Matrix Heatmap of the model after oversampling (DS-2) 53](#_Toc145280837)

[Figure 4. 14: T001 - Detects Neutral State Test 54](#_Toc145280838)

[Figure 4. 15: T002 - Detects Slightly\_Stressed State Test 54](#_Toc145280839)

[Figure 4. 16: T003 - Detects Very\_Stressed State Test 55](#_Toc145280840)

[Figure 4. 17: T004, T005, T006, T009 - Test results 55](#_Toc145280841)

[Figure 4. 18: T007, T010 - Test Results 56](#_Toc145280842)

[Figure 4. 19: T008 - Starts the timer test result 56](#_Toc145280843)

# LIST OF TABLES

[Table 1. 1: Research Gap 6](#_Toc145280706)

[Table 3. 1: Methodology Approach 14](#_Toc145280723)

[Table 3. 2: Tools and Technologies Used 15](#_Toc145280724)

[Table 3. 3: API Tests 44](#_Toc145280725)

[Table 3. 4: UI Test Cases 45](#_Toc145280726)

[Table 4. 1: API Test Results 49](#_Toc145304885)

[Table 4. 2: Model Test Final Results 54](#_Toc145304886)

[Table 4. 3: UI Test Results 58](#_Toc145304887)

[Table 4. 4: Summary of each student’s contribution 64](#_Toc145304888)

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| Abbreviation | Description |
| CSV | Comma Separated Value |
| ARIMA | Auto Regressive Integrated Moving Average |
| RNN | Recurrent Neural Network |
| SVM | Support Vector Machine |
| ML | Machine Learning |

# INTRODUCTION

Stress represents a significant challenge faced by IT professionals in their professional lives. As described in article [1], stress is defined as "the psychological and physiological reaction to an event or condition that is considered a threat or a challenge". It can arise from a variety of sources, including financial obligations, job loss, and workplace challenges, among others.

The proposed system introduces a novel approach, employing machine learning algorithms, to effectively detect and monitor the levels of stress and emotions experienced by the user and provide recommendations in assisting the user in relieving detected stress level and also assist in alleviating the emotion as well.

The system consists of 4 key components. They are,

* Keystroke dynamic based stress detection component
* Heart Rate Variability based stress detection component
* Face dynamic based emotion detection component
* Reinforcement learning based recommendation system component

The detected stress levels and the emotion outputs from the first 3 components will then be aggregated and it will output an aggregated stress level to the recommendation system which will then generate stress relieving recommendations and suggestions to the users based on their detected stress level to help them reducing and maintaining it in a low level. This approach facilitates early stress detection, enabling individuals to promptly act to maintain lower stress levels. This leads to heightened productivity and an improved quality of life. The report focuses on the implementation, approach, specifications, adaptability, and uniqueness of the keystroke dynamics-based stress monitoring component. This component not only represents a novel approach but also holds significant potential for practical application in real-world settings, particularly benefiting IT professionals facing high-stress work environments.

## **Background and Literature**

The concept of "stress" is a well-recognized and prevalent issue in society. It represents a significant health concern, with the potential to escalate from mild anxiety disorders to more severe conditions like burnout and heart diseases, which can ultimately lead to untimely fatalities. Consequently, experts and researchers have approached this issue in various ways. While numerous studies have delved into stress identification across diverse contexts, we have narrowed our focus to a select few studies that share common characteristics with the system component discussed in this report, namely, Stress Detection via Keyboard Dynamics.

In a notable research endeavor pertaining to stress detection via Keystroke Dynamics [2], a specialized keyboard equipped with a pressure sensor was employed [3]. This enabled the identification of stress through variations in keystroke pressure. Additionally, a capacitive mouse was utilized [4], and it was established that stress indeed exerts an influence on typing pressure, leading to observable variations. The study integrated the pressure variations from the keyboard with the frequency variations in mouse clicks for stress detection. Moreover, the researchers-maintained control over the users by providing specific instructions for the tasks to be performed in each session.

In a different approach, as detailed in reference [5], researchers have demonstrated how keystroke dynamics exhibit variations corresponding to different stress levels. The study involved collecting typing samples within a controlled environment where the stress levels of participants were manipulated. This manipulation included inducing stress through a multitasking user game, culminating in a negative comment on the user's performance, all with the intention of inducing stress.

In the study outlined in reference [6], researchers have identified a correlation between stress levels and variations in both keystroke and mouse dynamics. This relationship was examined through three distinct aspects: language familiarity, differing text lengths, and the imposition of time pressure. The findings indicate that users exhibit heightened stress levels when confronted with an unfamiliar language or when tasks require more time for completion. Furthermore, it was observed that elevated stress levels lead to an increase in keystroke latency, suggesting that individuals tend to type more slowly under stressful conditions.

In an alternative approach as described in reference [7], researchers have incorporated a pressure-sensitive keyboard to detect users' emotions. This is achieved by combining readings derived from keystroke pressure variations, dynamic time warping, and traditional keystroke dynamics. These three methods are integrated using a classifier fusion technique [8], resulting in a combined and enhanced output indicating the user's emotional state, which can range from anger, fear, happiness, sadness, surprise, to neutral and average.

In this approach [9] research objective is to identify early-stage stress levels by using keystroke analysis of gathered typing samples [10] gathered from a specific group of individuals exposed to various settings, including conditions of stress, high stress, and non-stress. The methodology employed in this study revolves around the examination of typing data collected from these participants, leveraging interactions with the keyboard.

The above discussion highlights the commonalities and limitations found in existing literature on detecting stress levels through keystroke dynamics. These studies typically employ pressure-sensitive keyboards, controlled experimental settings, artificial stress induction, and the collection of typing data from specific individuals. While these shared elements provide a foundation for research in this area, they also introduce certain constraints in terms of real-world applicability.

## **Research Gap**

Every research project has constraints and problems that need to be resolved in the future. This could be the result of not having enough time to do the study, not having the necessary technical and non-technical resources, technology advancements and new products introduced after the research was completed, as well as new trends and habits. In the literature that was thoroughly reviewed in the previous section there were some limitations that requires to be addressed,

In the approach discussed in [2], the emphasis on keypress pressure, while valuable, represents only one facet of keystroke dynamics. The reliance on a pressure-sensitive keyboard may limit its practicality for everyday use. Additionally, conducting the study in a controlled environment introduces the potential for biased results, as participants were provided with specific instructions, potentially influencing their behavior. The requirement for users to wear a bio sensor on their wrist further adds to the complexity and potential discomfort of the approach.

In [5], the artificial induction of stress holds promise for creating natural stress responses in participants. However, this method needs refinement before it can be seamlessly integrated into everyday use, as it currently necessitates a specific controlled environment for optimal results. Moreover, the reliance on physiological equipment to measure variables such as blood pressure, heart rate variability, and respiratory rate adds complexity and may not be conducive to widespread implementation.

The study in [6] successfully identifies variations in keyboard dynamics under stress. Nevertheless, the research acknowledges the need for further development in terms of implementation, including the calibration of systems and the creation of an automated output for user stress levels.

In the research discussed in [7], the identification of various emotions is a noteworthy advancement, but there remains room for further exploration. The reliance on a pressure-sensitive keyboard, however, could pose challenges in the implementation and adaptation phases, as acquiring such specialized hardware may be a prerequisite for achieving the desired results.

In the study outlined in [9], the effectiveness of the approach is evident, but concerns arise regarding the unsupervised data collection process. The absence of specific software or monitoring mechanisms to ensure data quality and integrity raises the possibility of participants providing inaccurate or falsified information, potentially leading to biased inputs for the researchers.

In summary the research gaps and limitations identified are as follows,

* The usage of a pressure sensitive keyboard to obtain keypress pressure may affect negatively in the implementation and the adaptation phase where the system is utilized in everyday use since the users are required to use a pressure sensitive keyboard to get the expected results and not every user would be able to acquire such equipment,
* Conducting all the experiments in heavily controlled environments which can lead to generating biased data inputs.
* Inducing stress artificially may generate actual stress or it may lead to incorrect readings.
* Gathering data that were collected in completely unsupervised environments may lead to incorrect data since there is no possible approach to validate the data gathered is accurate since it was entirely unsupervised.

The research component that is explained and discussed in this proposal report can,

* Detect users stress levels using any keyboard (external or integrated (laptop keyboard)) by taking various keystroke dynamics with the help of a program which runs in the machine background silently monitoring keystroke dynamics and feeding the gathered data to the trained model and acquiring the generated prediction related data.

This approach will,

* Eliminate the requirement of having a specialized keyboard.
* Eliminates the requirement of inducing artificial stress since there is no such requirement needed since the system is monitoring the user continuously.
* Utilizes trained machine learning model based on a supervised learning algorithm which automates the workflow.
* Acquire an aggregated stress level output and utilize it in base model retraining (incremental learning-based approach) to further finetune the trained machine learning model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Research**  **Features** | [2] | [5] | [6] | [7] | [9] | Proposed Component |
| Compatible with any keyboard | **Close with solid fill** | **Close with solid fill** | Checkmark with solid fill | **Close with solid fill** | Checkmark with solid fill | Checkmark with solid fill |
| Functional without various bio-sensors | **Close with solid fill** | **Close with solid fill** | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| Free from artificial stress induction | **Close with solid fill** | **Close with solid fill** | **Close with solid fill** | **Close with solid fill** | **Close with solid fill** | Checkmark with solid fill |
| Adaptable to everyday use | **Close with solid fill** | **Close with solid fill** | **Close with solid fill** | **Close with solid fill** | **Close with solid fill** | Checkmark with solid fill |

Table 1. 1: Research Gap

## **Research Problem**

Staring at a computer screen for long hours is an extremely unhealthy task for an individual to be carrying out. As discussed in earlier sections stress is a common problem all IT professionals face. This can be due to isolation from co-workers with the increasing utilization and adaptation of the work from home option or it can be since [11] the workload is extremely high, tight deadlines, and it requires them sitting in the same position for long periods of time staring at the computer screen.

While IT industry is recognized to be one of the most flexible job industries in the world it does has its pros and cons. In this case occupational stress has become a challenging issue for the IT industry as the number of IT professionals who experience stress is increasing rapidly.

As mentioned in the earlier paragraph some of the reasons could be due to the isolation created by the work from home methods or it could be the excessive workload the individual receives. Either way stress is an issue that needs to be dealt with immediately as discussed in earlier sections.

The main research problem that is addressed by the complete research project is,

“How to detect users stress levels and emotions without interrupting or impacting the user’s day to day work by utilizing the simple equipment they use everyday and assist in relieving the detected stress levels and alleviate the detected emotion?”

In order to properly address the identified research problem, 4 sub research problems were derived and addressed via 4 individual components. The derived research problem that fuels the component that is discussed in this report is,

“**Can keystroke dynamics be utilized in detecting stress levels of individuals? Keystroke dynamics is a unique factor that varies from person to person therefore, how can one detect each individuals’ stress levels?”**

The problem was simplified and as it explains since keystroke dynamics vary from user to user it is not practical to detect stress levels by using a constant approach. Therefore, the component was developed using a dynamic approach allowing the component to learn continuously and adapt its detection by learning each user typing patterns and keystroke dynamics by analyzing new data.

Based on the identified research problem the component should address the following factors,

1. Detect and log the keystroke dynamics of the user without effecting their privacy or day to day activities.
2. Should be able to predict the stress level of the user by analyzing the gathered keystroke dynamic data.
3. Should be able to improve its predictions and be able to learn from new data continuously to better suit each user who is using the application.

The approach and its properties, functionality, implementation will further be explained in detail in the upcoming sections of the report.

# RESEARCH OBJECTIVES

## **2.1 Main Objectives**

The main objective of the proposed system is to design and develop a novel system where it could detect and determine stress levels and emotions of the user and assist them in relieving the detected stress levels and alleviate the emotion using the system with the help of 4 machine learning based components,

* Component 1: Keystroke dynamic based stress detection.
* Component 2: Heart Rate Variability based stress detection.
* Component 3: Facial dynamics-based emotion detection.
* Component 4: Re-enforcement learning based recommendation system.

Each component is responsible for its separate main objective and contributes towards addressing the overall projects research problem. Component 1 will be analyzing the keystroke dynamics and predicting stress levels, component 2 will be analyzing the HRV of the user and predicting the stress levels, component 3 will analyze facial dynamics and detect the emotion of the user, component 4 will receive an aggregated result of the current stress level and the emotion of the user from the components 1, 2 and 3 to generate and provide recommendations to the users to assist them in relieving the detected stress levels and emotion.

The overall system will be installed to the user’s machine as a simple desktop application which will run in the background allowing the users to freely carry out their work.

The main objective the component discussed in this report will be addressing is,

**Keystroke dynamic based stress detection. (Detecting stress levels of users by analyzing users keystroke dynamics)**

## **2.2 Specific Objectives**

The main objective which is addressed by the component that’s discussed in this report was split into smaller sections and was addressed via 4 sub objectives.

1. Train a machine learning model, finetune it, configure the data communication of the model outputs and inputs to predict the stress levels of users by analyzing their keystroke dynamics.
2. Develop a program to acquire keystroke dynamics of the user in 20-minute sessions and log each session data in CSV files.

* Should support real-time keylogging
* Should support on-demand keylogging

1. Acquire the aggregated stress level of the current users’ session and prepare the dataset for incremental model training.
2. Prepare the complete compatible dataset and set up a job to retrain the model in cycles of 24-hour time periods.

The above 4 sub-objectives contribute towards completing the component functionality. The objectives and their development, implementation and results will be further discussed in the methodology section when the component functionality is explained and illustrated.

# METHODOLOGY

This section comprises three sub-sections that explain the methodology of the entire system. These sub-sections discuss and illustrate the approach, the implementation and the testing of the product and the commercialization approach of the product,

# Methodology

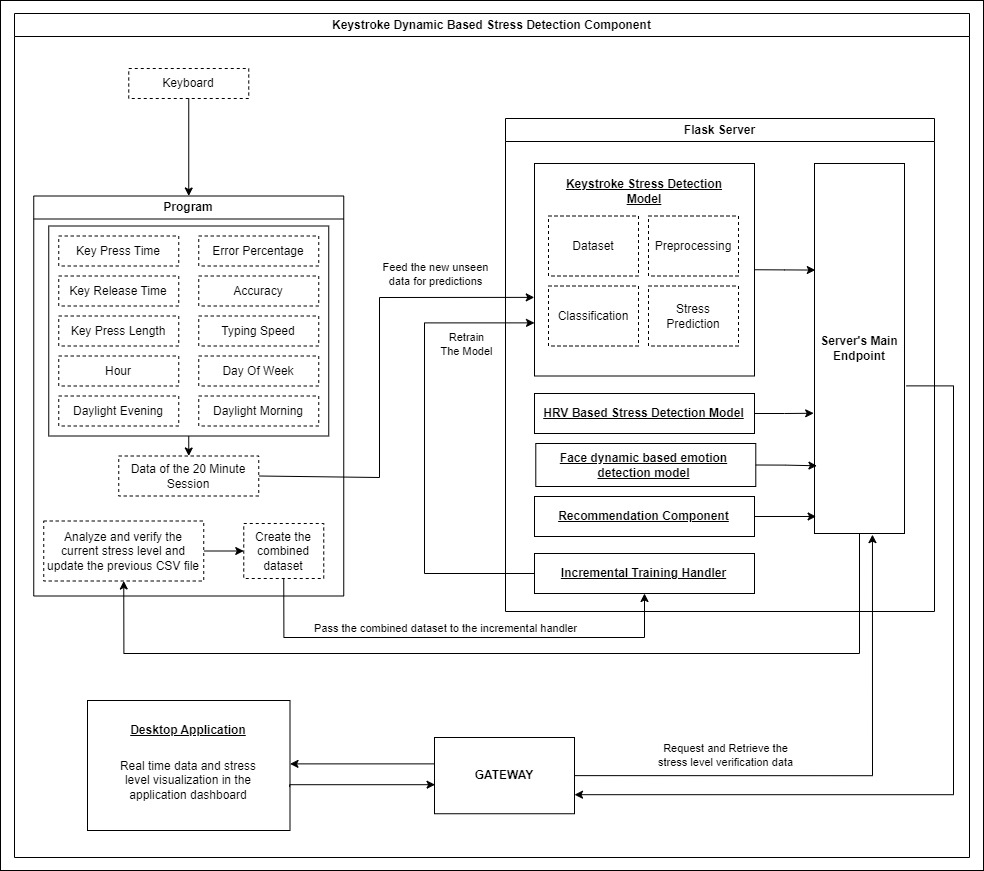
As discussed in the above sections the overall project is a combination of 4 main components where each of them has its own contribution towards the overall project. However, this section will focus on illustrating the methodology of the **keystroke dynamic based stress detection component.**

### Data Collection

The collection of data was conducted via various platforms such as Kaggle, WhatsApp, Microsoft Teams, Zoom, Microsoft/Google Forms etc.… For model training various options were considered but finally the training data set [12] was acquired from Kaggle. When considering about stress related discussions and brainstorming sessions were conducted via Microsoft Teams, Zoom, WhatsApp. Data relating to views and ideas on certain aspects on stress and the project were acquired via Microsoft / Google Forms. The new unseen data (keystroke dynamics of the user) was gathered via the developed python program and the data does not acquire the key labels for privacy reasons it was ensured that no such data was gathered.

### Approach

This section will discuss and emphasize on the approach that is underlying this entire project.



*Figure 3. 1: Keystroke dynamic based stress detection component diagram*

As per the illustration of the component workflow diagram the basic process is as follows. The user will be using the keyboard and the program that is developed will be running in the background. When the user starts typing on the keyboard it will start a timer of 20 minutes and start logging the users keystroke dynamics into a CSV file and it will send the completed CSV file to the ML model hosted in the server when the 20-minute timer is completed.

The model will then analyze the data and predict the current stress level of the user and send the level to the application. Within this same time the HRV based stress detection component and the facial dynamic based emotion detection component will also be sending the prediction values generated from their models to the application as well. These 3 prediction outputs will then be combined together and output an aggregated stress level to the application.

This aggregated stress value will also be acquired by the keystroke dynamic based stress detection component and will be appending the previous CSV file with the stress level data making it compatible for model retraining. System will merge this files data to the combined dataset CSV file. This will be considered as one session and this process will continue as 20-minute sessions until the user stops the work.

Finally, after a 24hour timer is completed the combined dataset will be used to retrain the current base model of the system and the process will continue the same way.

The keylogger program should be developed so that it supports both real-time keylogging as well as on demand keylogging since this will also be integrated with the application as well. The keylogger should be activated whenever the system is running and it should not intervene with the user’s day to day activities

Below is a high-level breakdown on how the methodology was applied to each of the derived sub objectives to complete the required task,

|  |  |
| --- | --- |
| Requirement | Applied Approach |
| Train a Machine Learning Model | Gather a dataset, train a few machine learning models and test them and select a suitable ML model for the task. |
| Develop a program to acquire keystroke dynamics of the user in 20-minute sessions and log each session data in CSV files | Develop a program using python language by incorporating the logic required. |
| Configure the data communication of the model inputs and outputs | Host the ML model in a Flask server and configure communication via REST API endpoints. |
| Acquire the aggregated stress level of the current users’ session and prepare the dataset for incremental model training. | Fetch the aggregated stress levels from the application and write a function using python language to handle the incremental model retrain logic. |
| Prepare the complete compatible dataset and set up a job to retrain the model in cycles of 24-hour time periods. | Write a function using python language and schedule a job to run the function with a timer. |
| Realtime visualization in the desktop application | Integrate the backend functionality to the Desktop Application. |

Table 3. 1: Methodology Approach

### Tools and Technologies

The table presented below offers a comprehensive account of the tools and technologies employed in the development of the application as well as the program.

|  |  |
| --- | --- |
| **Description** | **Tools and Technologies** |
| Programming IDE | Visual Studio Code |
| Programming language for desktop application development | React JS, Electron JS |
| Machine learning algorithm-based Keystroke dynamic based stress detection model. | Python language with Google Collabotary |
| Programming language for backend development | Node JS |
| Database for store data | Mongodb |
| Hosting the API | Flask |
| Version Controlling | Gitlab |
| Team connectivity | Teams and WhatsApp |

Table 3. 2: Tools and Technologies Used

# Commercialization Aspect of the Product

The commercialization strategy is a critical component of our project, as it outlines the pathway to bring our innovative solution to market and make it accessible to a wider audience.



Figure 3. 2: Commercialization Discussion

**Target Audience**

Central focus for the commercialization strategy will be on corporate professionals, particularly in the IT sector, who are known to experience elevated stress levels. The application will be marketed to both IT companies and individual IT professionals. Additionally, we plan to extend our offering to college students in the near future, as they represent another demographic with heightened stress levels.

**Revenue Generation**

To generate revenue, we will adopt a subscription-based model, providing different pricing tiers based on the number of users and the level of functionality they require. We will also explore corporate partnerships, offering access to the application for all employees at a discounted rate. In addition to subscription fees, we will leverage in-app advertising and establish partnerships with stress-reducing product companies to further diversify our revenue streams.

**Marketing Strategy**

**Phase 01:**

Launch the initial version of the software in a controlled environment, specifically at an IT services company, to obtain valuable feedback and reviews.

**Phase 02:**

Introduce a complimentary version of the software with a limited set of activities, alongside a professional version that unlocks full features based on subscriptions.

During this stage, the main focus will be on IT and Software companies, highlighting the advanced functionalities of the professional version.

**Phase 03:**

Execute targeted marketing campaigns through online advertising, utilizing social media platforms, and participating in industry conferences to connect with potential clients. Furthermore, we will partner with HR departments and employee wellness programs to position the software as a vital tool for managing stress and enhancing employee well-being.

**Phase 04:**

Collect input from clients and consistently incorporate updates and improvements into the software based on their feedback and evolving requirements. This method will not only ensure high levels of customer satisfaction and retention but also draw in new clients through positive word-of-mouth referrals.

**Phase 05:**

Consider the possibility of forming partnerships with health insurance providers to provide our software as a wellness benefit to their clients. This initiative has the potential to not only generate a fresh revenue stream but also make the software more accessible to a broader audience

# Testing and Implementation

This section is divided into two subsections: implementation and testing. The implementation section explains the development process. The testing section contains a list of some of the test cases used to test the program.

### Implementation

The implementation phase consists of all the details from model selection to the application testing part of this component.

**Machine Learning Model Development**

The gathered dataset [12] from Kaggle was analyzed and it contained both timeseries related qualities as well as classification related qualities.

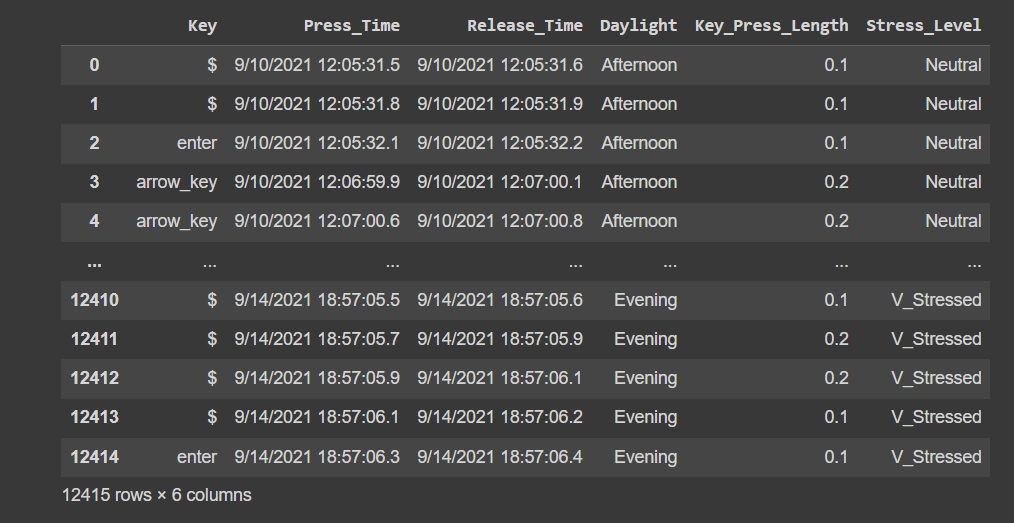


Figure 3. 3: Dataset Before Preprocessing

As illustrated in the figure it consisted of 12415 rows of keystroke dynamics related data under the following columns,

* Key
* Press\_Time
* Release\_Time
* Daylight
* Key\_Press\_Length
* Stress\_Level

Also there was another dataset with the same columns and it consisted of 14818 rows.

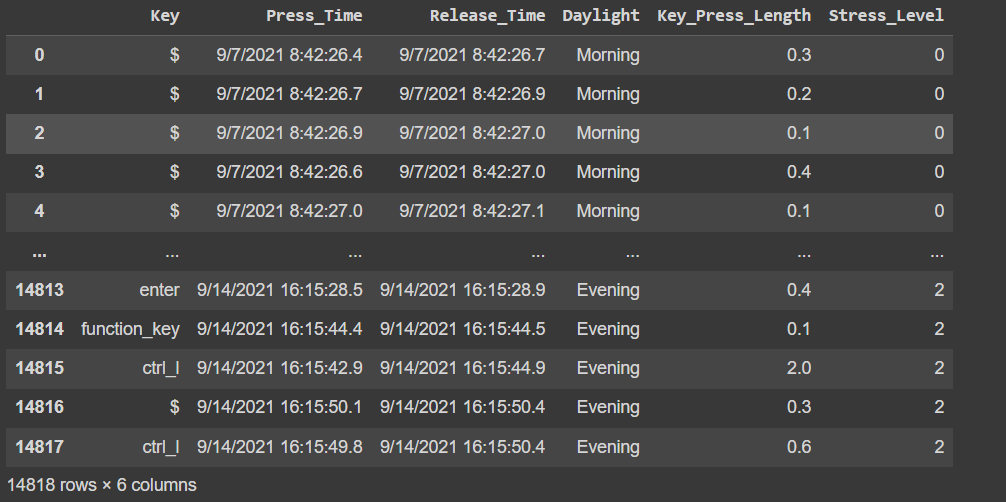


Figure 3. 4: Dataset 2 Before Preprocessing

The end goal of the model was to analyze the keystroke dynamics and predict the stress level. Since the dataset had 2 different aspects it was trained using both types of models.

1. Timeseries Models
   1. Long Short-Term Memory Model
   2. ARIMA Model
2. Classification Models
   1. Support Vector Machine Model
   2. Random Forest Model

The dataset was first preprocessed using preprocessing techniques before proceeding with the model training. The Stress\_Level column was first converted into Int values using Label encoding techniques.

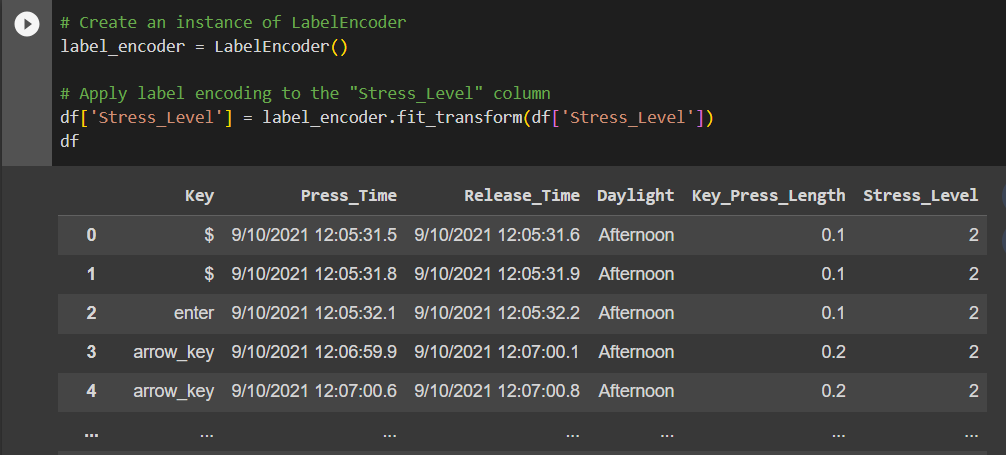


Figure 3. 5: Label Encoding on Dataset

Afterwards, from the Press\_Time and Release\_Time the Hour of the day and the day of the week was derived.

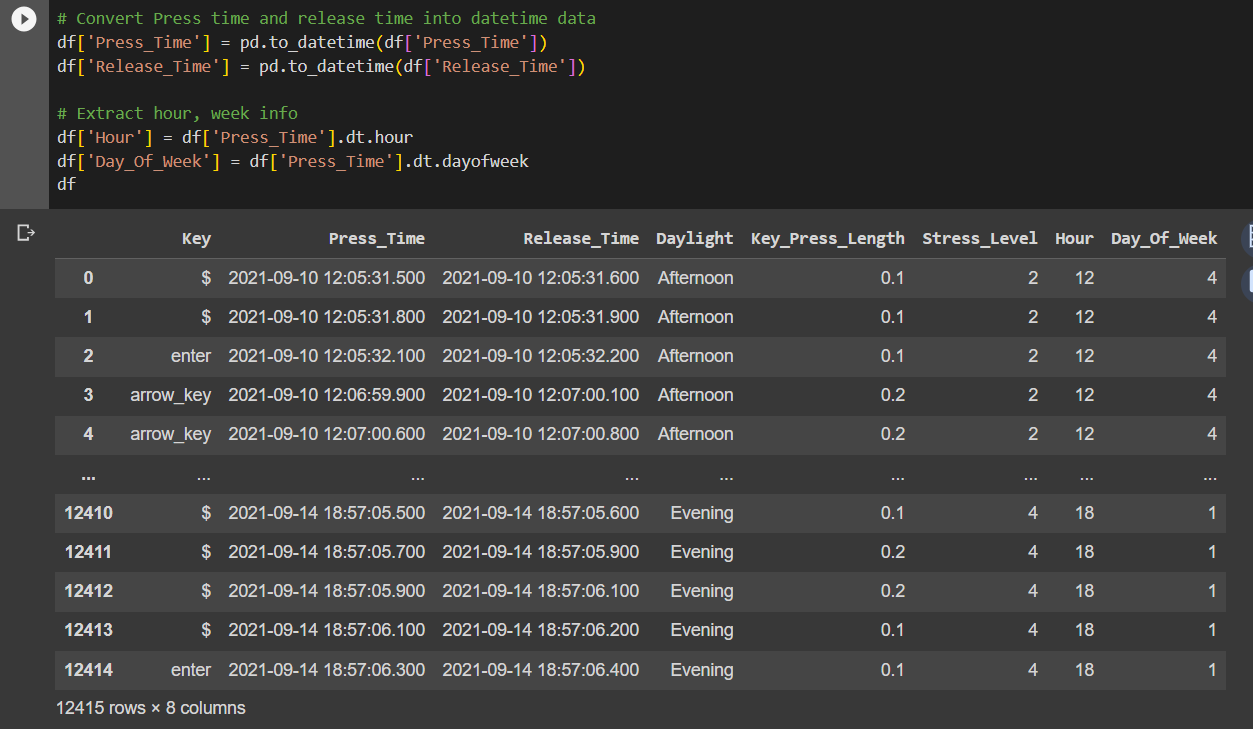


Figure 3. 6: Extracting Hour and Day of Week data

Next, one hot encoding was applied to the Daylight column and derived the Daylight\_Morning and Daylight\_Evening columns.

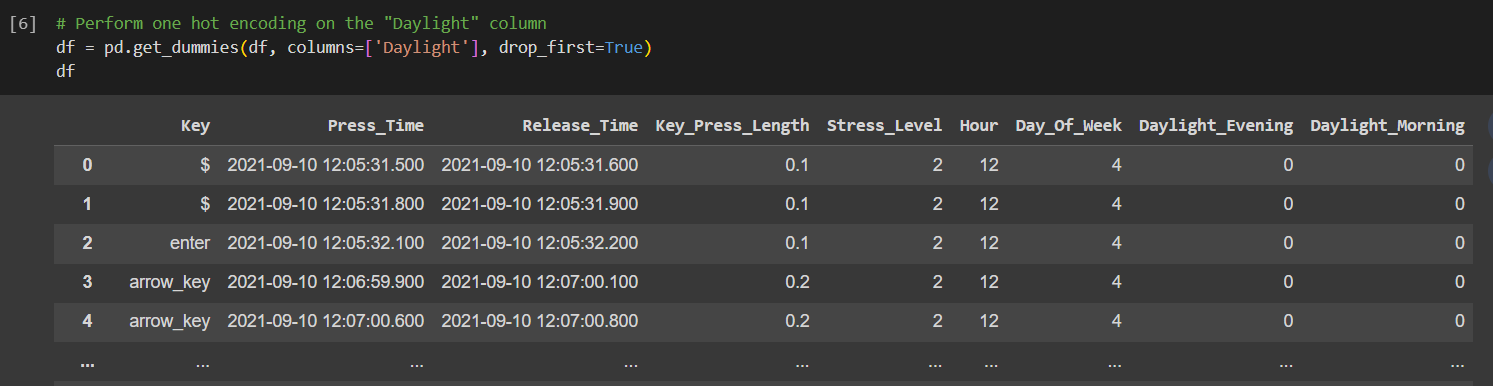


Figure 3. 7: One Hot Encoding on the dataset

Now the basic preprocessing was completed and was ready to train the models.

RNN LSTM model was trained with the dataset which contained 14818 rows and the first tests were conducted only by selecting the target variable and allowing it to learn the pattern. The test and train sets were split accordingly,

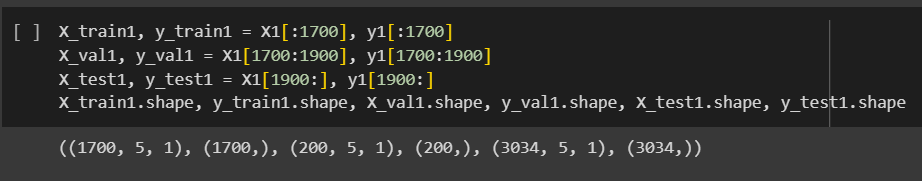


Figure 3. 8: RNN LSTM Train Test Split

The model was able to produce good predictions. However, the target variable was visible during the training phase and it only contained the target variable. The prediction of the component should be based on the acquired keystroke dynamics of the user. Therefore, the model was tested again with other features. Unfortunately, the model was not able to successfully predict the stress level instead it was only able to predict one stress level for the whole test set. This could have been a reason with the dataset. It requires a considerable amount of data with a considerable amount of variations in order to generalize properly to new variations and patterns that occur in the new data. Therefore, the model was dropped from the selection.

Next model was the ARIMA model and it was also trained with the dataset which contained 14818 rows and was split into test and train sets as follows,

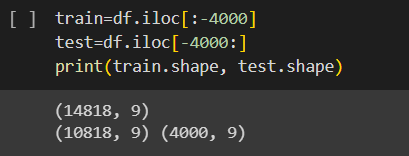


Figure 3. 9: ARIMA Train and Test Set

This time without making the target variable visible or removing the other features the complete preprocessed dataset was fed after splitting the test and train sets. Unfortunately, the prediction results were not accurate and only predicted one state of the stress levels just like the earlier model. The target variable is a categorical variable and even though the dataset contains a dependency between the time data and the stress level. Based on the results of the above two models it seems that the dependency is low. Therefore, the ARIMA model was also dropped and proceeded to train the Classification Models.

The SVM model was trained using the same 14818 row dataset and the train and test split was done as follows,

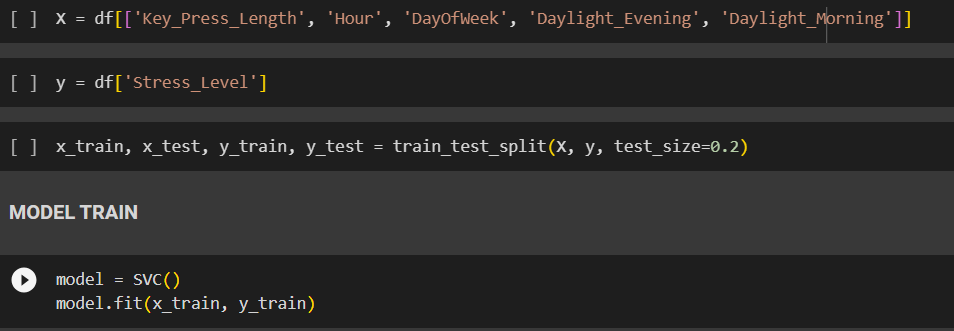


Figure 3. 10: SVM Model Train and Test Set

The model was trained with the above features and the Stress\_Level was set to be the target variable. After training the predictions were tested with the test set and it managed to predict somewhat accurate results and the accuracy score was at 0.5954790823211876. This accuracy level and the predictions were better than the previously tested both Timeseries models and the Classification approach seem to be accurate.

Next model that was tested during this model selection phase was the Random Forest Model. Same as before the test and train split was performed as follows,

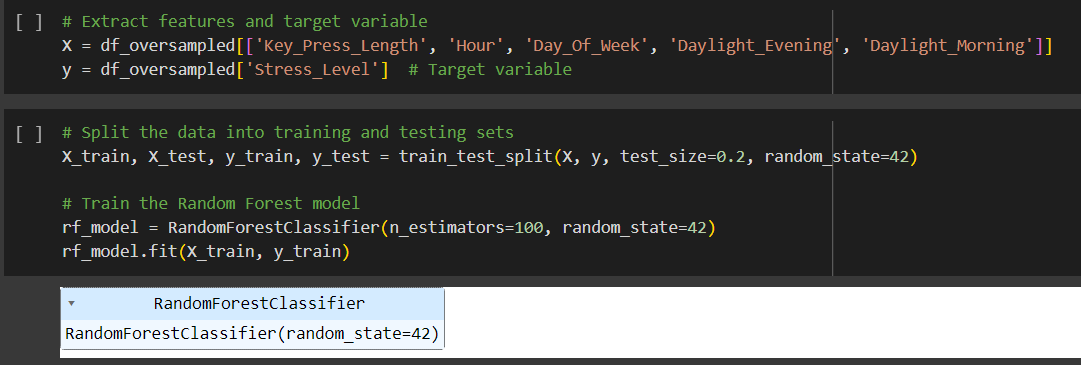


Figure 3. 11: Random Forest Train and Test Split

The model was trained using the 14818 row dataset and the predictions significantly improved. Used n\_estimators as 100 so that it will perform better when generalizing to new unseen data. The accuracy level was placed at 0.7844129554655871 and this is the highest accuracy that was achieved out of the models trained. With this in the model as the best model for the dataset and the second model was the SVM model based on accuracy and prediction.

Before finalizing the model to be used as the base model the dataset was balanced out using an oversampling technique and tested with both datasets on both SVM and Random Forest models



Figure 3. 12: Before Oversampling



Figure 3. 13: Oversampling

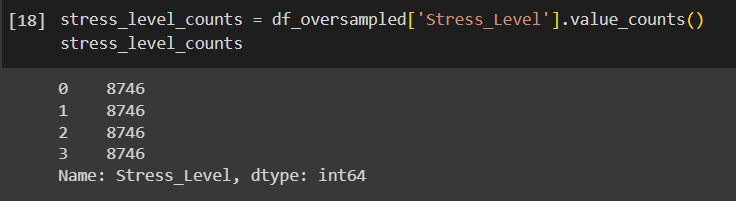


Figure 3. 14: After Oversampling

The models were tested and the Random forest model received an accuracy of 0.8230670287265971 while the SVM model received an accuracy of 0.7144490495926826. Therefore, the Random forest model was selected as the base model for the component. Evening

Now before finalizing which dataset to be used another test was carried out with the 12414 row dataset. It went through the same preprocessing techniques and also the oversampling technique as well. The accuracy was placed at 0.8359335681932561 and the data set and the model was finalized,

* **Model** : Random Forest
* **Accuracy** : 0.8359335681932561
* **Precision** : 0.8487462576667454
* **Recall** : 0.8359335681932561
* **F1** **Score** : 0.8363306604335913

The finalized dataset is as follows,

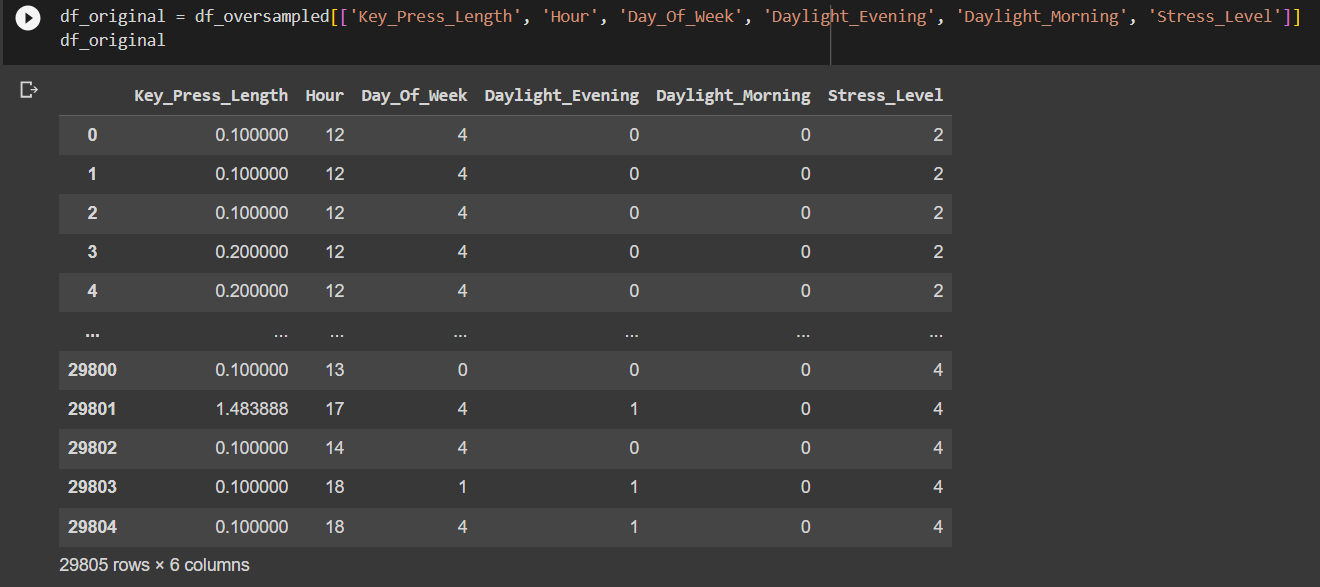


Figure 3. 15: Finalized Dataset

The selected feature set for the training was,

* Key\_Press\_Length
* Hour
* Day\_Of\_Week
* Daylight\_ Evening
* Daylight\_Morning
* Stress\_Level

This concludes the implementation of the Model Development Process

**Keylogging Program Development**

After the model was finalized the next objective was to develop a program to acquire the users keystroke dynamic related data to feed into the model for predictions. Therefore, a Keylogging program was developed using python programing language. The program serves 2 main purposes,

* Realtime Keylogging
* On-Demand Keylogging

During 20-minute cycles the program will save the CSV file with the keystroke data and send the file to the model for predictions.

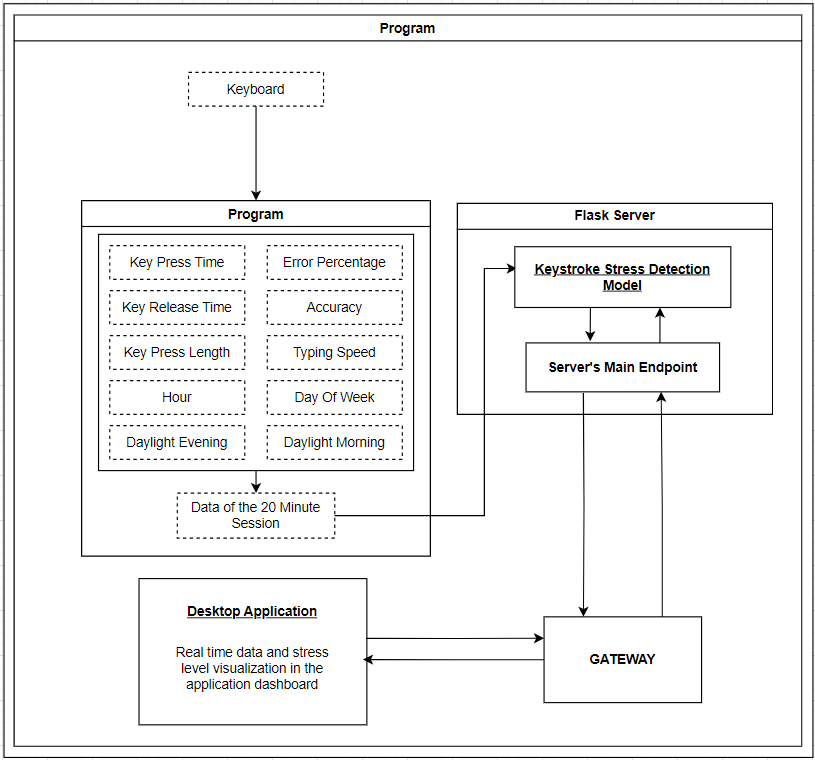


Figure 3. 16: Keylogger Functionality

The program will be gathering the following metrics of the user’s session,

* Key press time
* Key release time
* Key press length
* Day of week
* Hour
* Daylight morning
* Daylight evening
* Error percentage
* Accuracy percentage
* Typing speed

And these data will be recorded in a CSV file and the file name will be the timestamp of the start time of the recording. “data\_20230906220243.csv”.

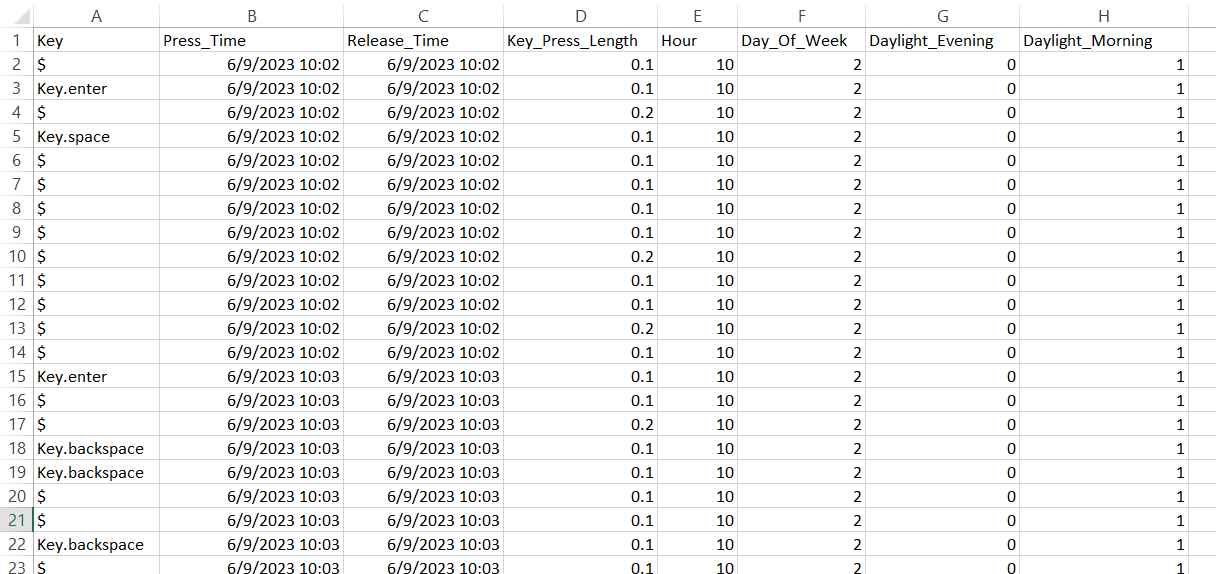


Figure 3. 17: Gathered keystroke dynamics snapshot

The program employs a comprehensive set of metrics derived from keystroke dynamics to assess user performance. These metrics include:

* **Key Press Length (KPL)**: Calculated as the difference between the Release Time (RT) and Key Press Time (PT).
* **Error Percentage (EP):** Computed from the ratio of Total Characters (TC) to the Number of Backspaces or Delete Key Presses (BS), expressed as a percentage.
* **Accuracy (A):** Represented as 100 minus the Error Percentage (EP), providing a measure of typing accuracy.
* **Session Time (ST):** Determined by the difference between the Current Time (CT) and the Cycle Start Time (CST) of the session, converted from seconds to minutes.
* **Typing Speed (TS):** Quantified in Words Per Minute (WPM) and derived from the Word Count (WC) of the session divided by the Session Time (ST).

Additionally, the program leverages date and time information derived from keypresses to generate additional contextual metrics, including:

* **Hour (H):** Reflecting the hour of the day.
* **Daylight Evening (DE):** Indicates whether it is evening.
* **Daylight Morning (DM):** Indicates whether it is morning.
* **Day of Week (DOW):** Represented as a numerical value ranging from 0 to 6, where Sunday is denoted as 0 and Saturday as 6.

The below equations were utilized in gathering the metrics,

* KPL = RT – PT
* EP = (Σ(TC) / Σ(BS)) \* 100
* A = 100 – EP
* ST = (CT – CST) / 60
* TS = Σ(WC) / ST

To safeguard user privacy, the program employs a symbolic representation of alphanumeric key labels, ensuring that the actual content remains confidential. This measure helps prevent sensitive information from being disclosed during the analysis process.

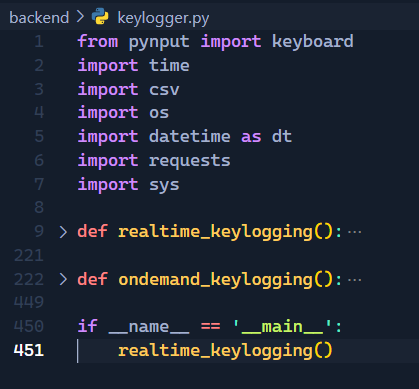


Figure 3. 18: Keylogger Functions

This concludes the basic Keylogger Development process. This will serve the basic functionality which is keylogging and saving to CSV file.

**Configure Data Communication Between Model Inputs and Outputs**

In order for the trained ML model to communicate with the program and the application it should be hosted in a server where it allows communication using API endpoints. Therefore, a dump of the ML model is generated and hosted in a FLASK server.

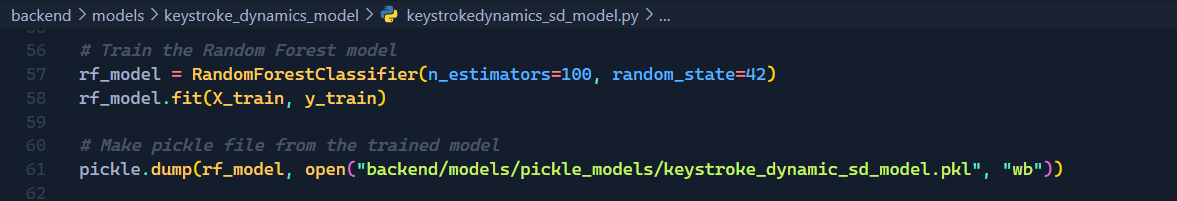


Figure 3. 19: Generating dump file of the ML model using pickle library

Once the model is hosted in the server the communication can be conducted by defining REST API endpoints.

Now the keylogger is able to send the generated CSV file to the ML model.

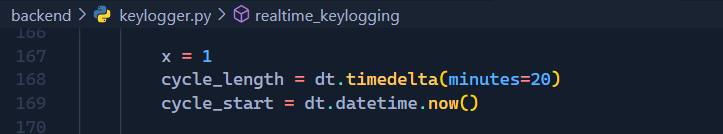


Figure 3. 20: Session timer

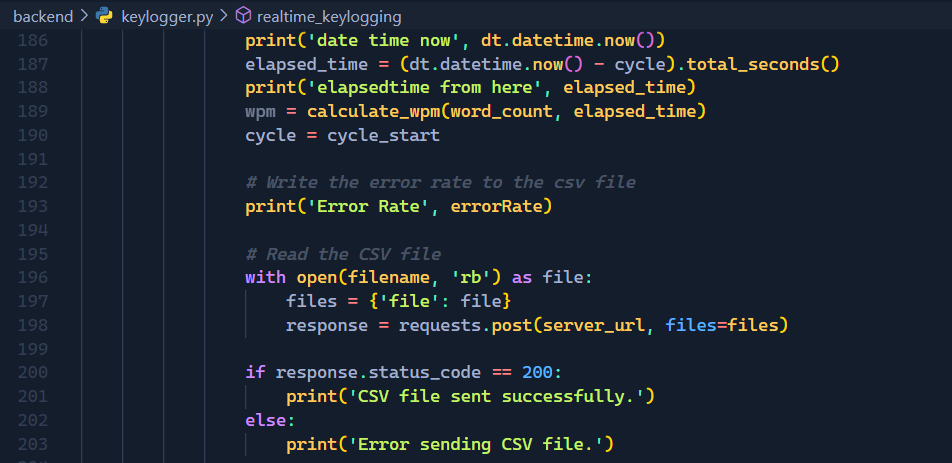


Figure 3. 21: Send Generated CSV file to the ML model code snippet

Once the CSV file is received by the FLASK server and the REST API endpoint will be running a function to trigger the prediction of the model. The model dump that is stored in the FLASK server will be loaded and that instance will be used as the model.

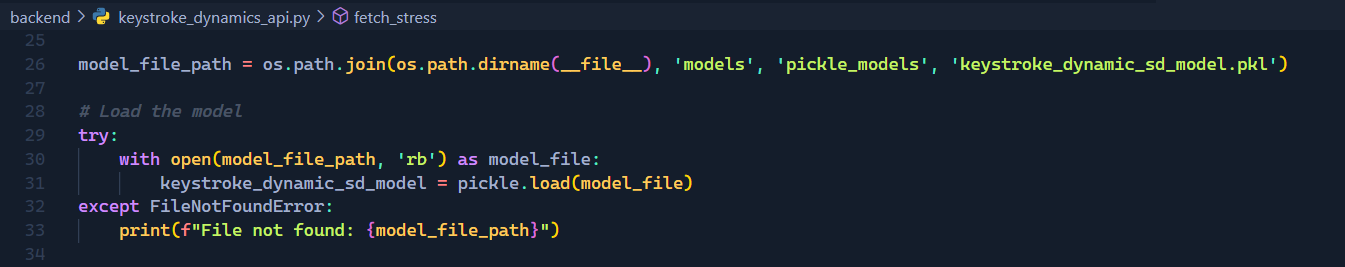


Figure 3. 22: Loading model dump from server

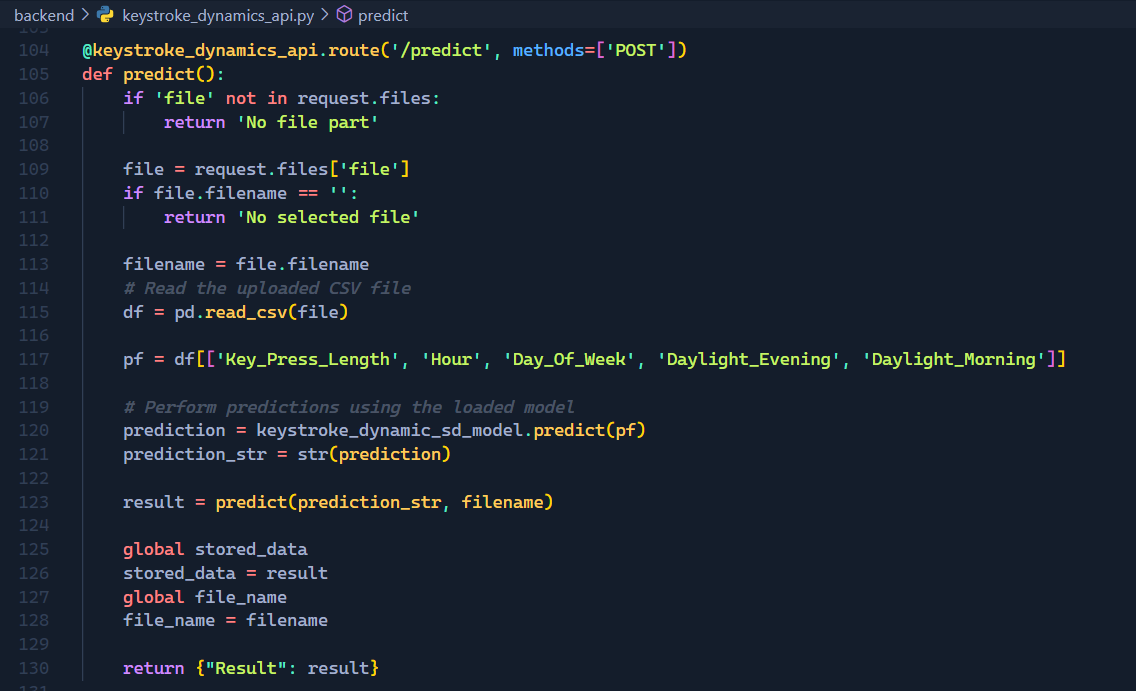


Figure 3. 23: Prediction function

The prediction result will be acquired by the application. Before that the output will be set to another API endpoint.

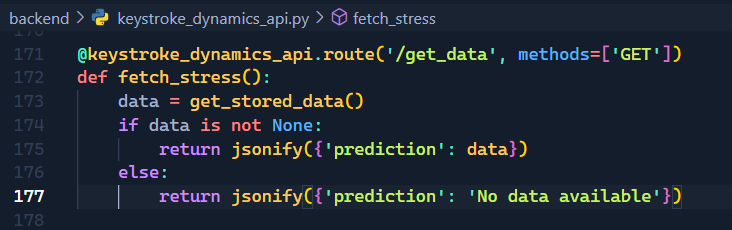


Figure 3. 24: Prediction result fetch function

The fetched result will be aggregated with the other 2 prediction outputs that were acquired by the HRV stress detection component and the Facial dynamic based emotion detection component. This will generate an aggregated stress level based on the output combinations.

The data communication between the model and the program and the application was configured.

**Configuration of Stress Level Aggregation and CSV file Preparation**

The aggregation was done on a hypothesis and this was drawn after the consultation of the external supervisor Mrs. Arosha Dasanayake. The hypothesis used for the aggregation of prediction outputs is as follows,

1. The system is **not** measuring **Clinical Stress**
2. Measure the stress level on a **scale of 1 to 10**
   * **Neutral (not stressed): 1-2**
   * **Slightly Stressed: 3-4**
   * **Stressed: 5-7**
   * **Very Stressed: 8-10**
3. These values will be assigned to the combinations received from the 3 components.

Eg: The combination,

* **Slightly\_Stressed – stress – Angry** would be considered as **Stress Value of 9** on the scale.
* Therefore, the aggregated stress level would be **Very Stressed.**

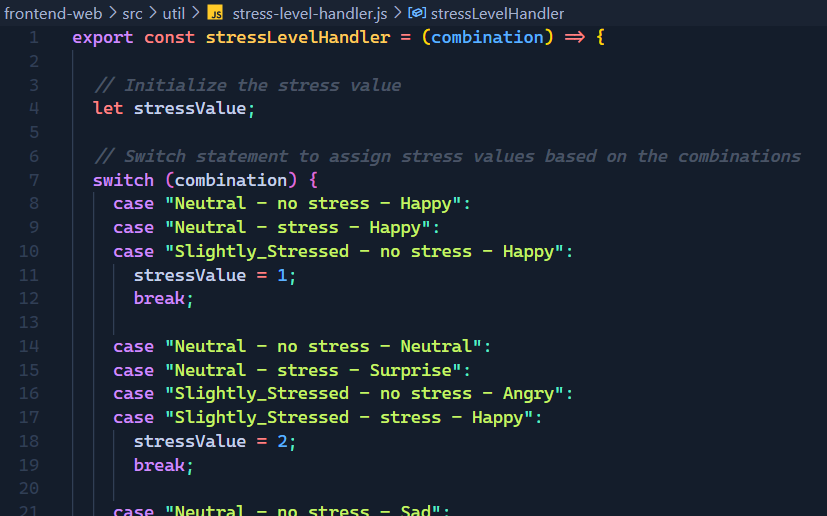


Figure 3. 25: Stress Level Handler code snippet for handling combinations



Figure 3. 26: Stress Level Handler code snippet for final output

After the combination is acquired it will be used in the application. Meanwhile another API call will be sent from the FLASK backend to retrieve the aggregated stress level.

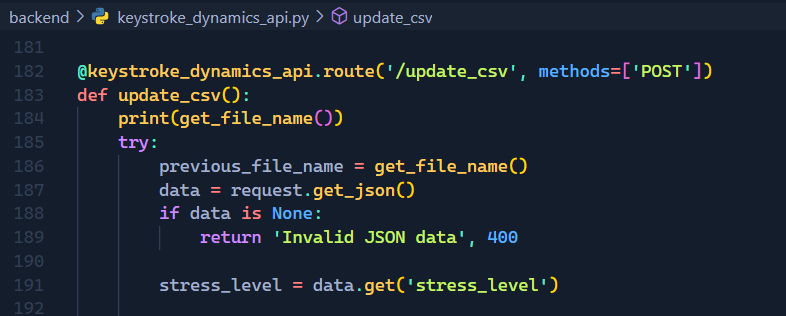


Figure 3. 27: Retrieve Aggregated Stress Value

Once the aggregated stress level is received by the API the function update\_csv is triggered and it will be utilized in appending the previously created CSV file which was responsible for the stress level prediction that was included in the aggregated stress level.

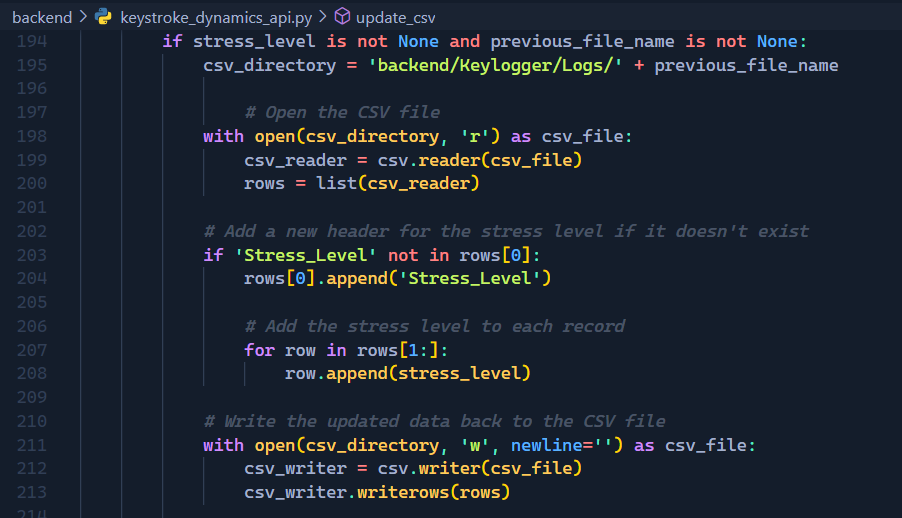


Figure 3. 28: Stress level appending code snippet (continuation of update\_csv function)

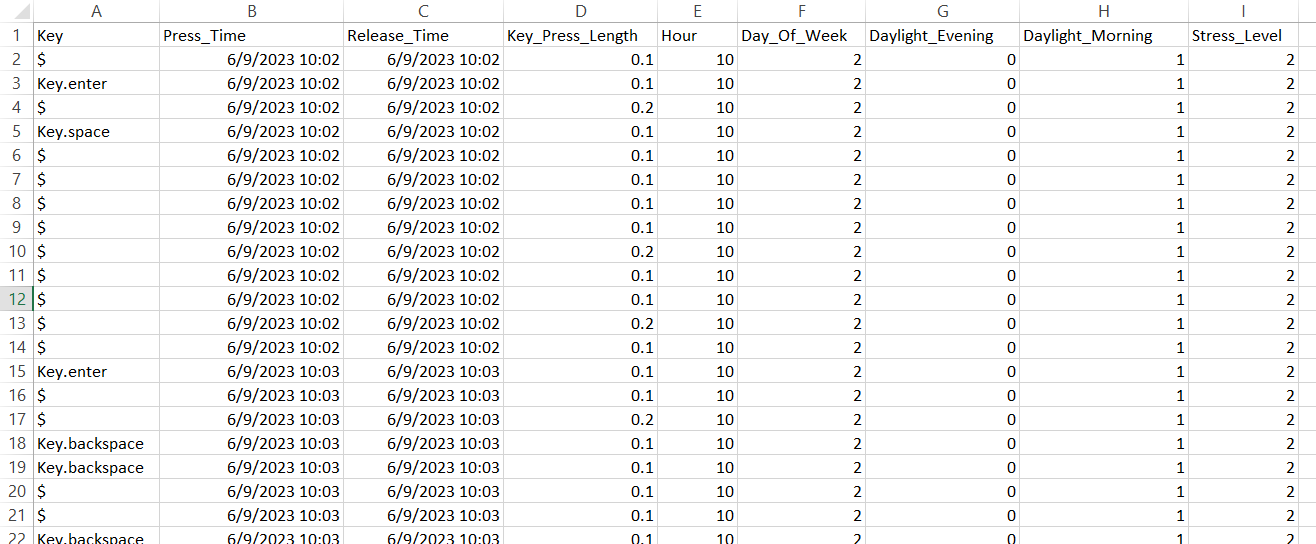


Figure 3. 29: After appending the Aggregated Stress Level

Also, after appending the CSV file with the stress level data it will be compatible to be used for model retraining since it contains all the necessary data in the appropriate format to be utilized in the model retraining process. Therefore, this entire file will be appended to a copy of the preprocessed dataset that was used in training the model.

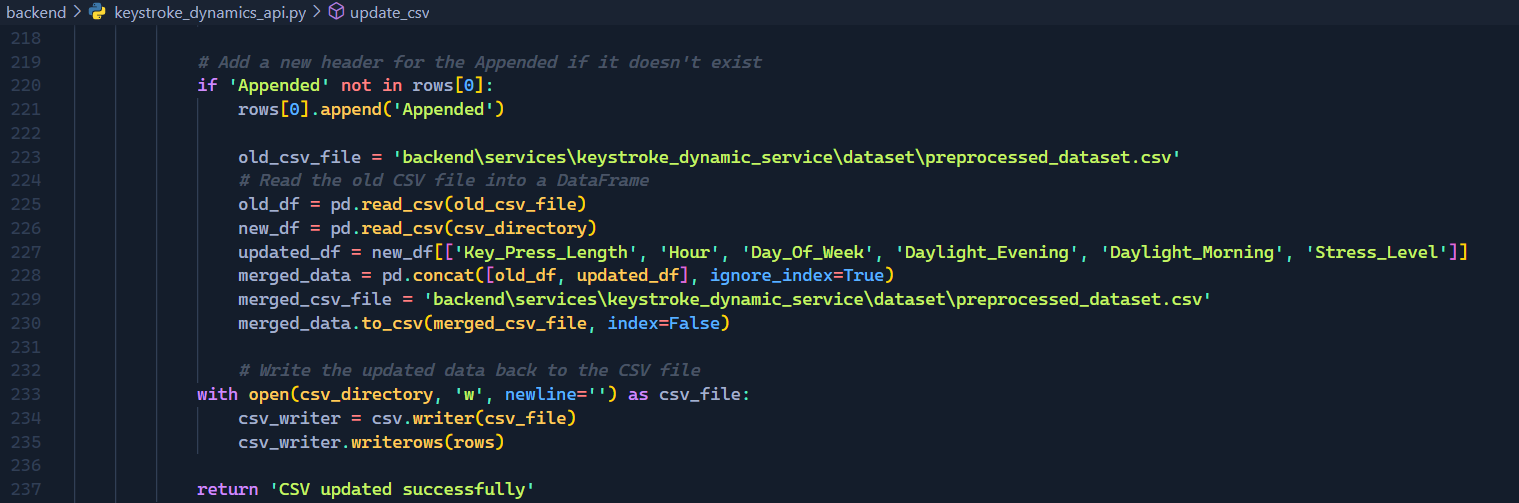


Figure 3. 30: Append the file to the combined dataset file

Once the file has been appended to the combined dataset the CSV file will get a new header called Appended. This will prevent the same file from being merged to the combined dataset more than once.

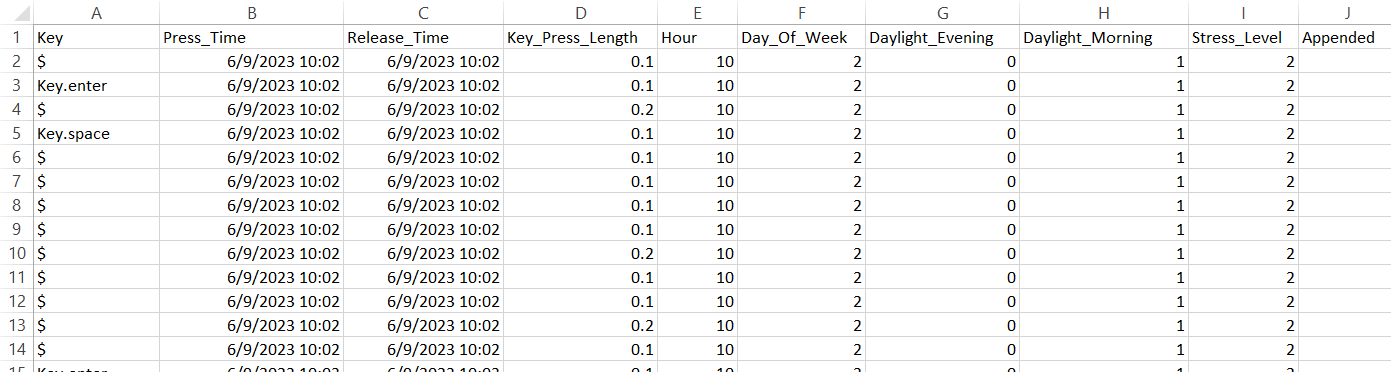


Figure 3. 31: CSV file after it has been appended to the combined dataset

Eg: *A hypothetical scenario explaining the process****.***

The CSV file *data\_20230907110940.csv* has been generated on the 07th of September 2023 at 11:09:40 so after the 20-minute session this file has been sent to the model for predictions that would be at 11:29:40. This contains the data from 11:09:40 to 11:29:40 so the prediction would be based on that time frame. Let’s assume that the stress level prediction output was “Neutral”. During this same time the other 2 component outputs will also be retrieved. Let’s assume their predictions were “no stress” and “Happy” then the stress level value would be 1 according to **Figure 3.25**. This means the stressLevel is “**Neutral**” according to **Figure 3.26**. This is the aggregated stress level for the time period 11:09:40 to 11:29:40 and this will be retrieved by the backend and the **update\_csv function** will append this aggregated stress level to the CSV file *data\_20230907110940.csv*. Afterwards, this file data will be appended to the copy of the preprocessed dataset “preprocessed\_dataset.csv” that was used for model training. This process will occur in every 20-minute session. All these tasks will happen within seconds and during this time another CSV file will already have been generated and it will be logging the current keylogging values.

The Aggregated Stress Level was acquired and the CSV file was prepared for the incremental model training.

**Incremental Model Retraining Configuration**

Once the CSV files have been appended as explained in the above section after each 20-minute session the combined preprocessed dataset will also be appended accordingly. As discussed in the above sections this approach is an incremental approach where the base model should be able to learn new data and increase the accuracy of the predictions since the keystroke dynamics is unique for each user.

The prepared dataset will be used for the incremental model training and a function was defined to handle this approach,

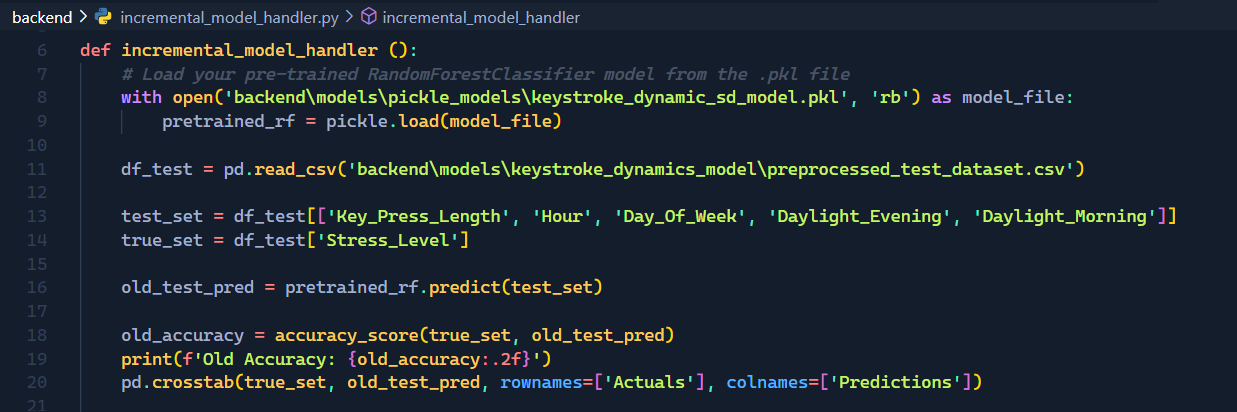


Figure 3. 32: Incremental Model Handler loading model and testing current accuracy code snippet

The incremental model handler will load the current model hosted in the server from the dump file and will also load a test dataset for testing the accuracy of the model. Before the model retraining the current model, accuracy will be tested and recorded. Then the combined preprocessed dataset will be loaded and utilized in retraining the model.

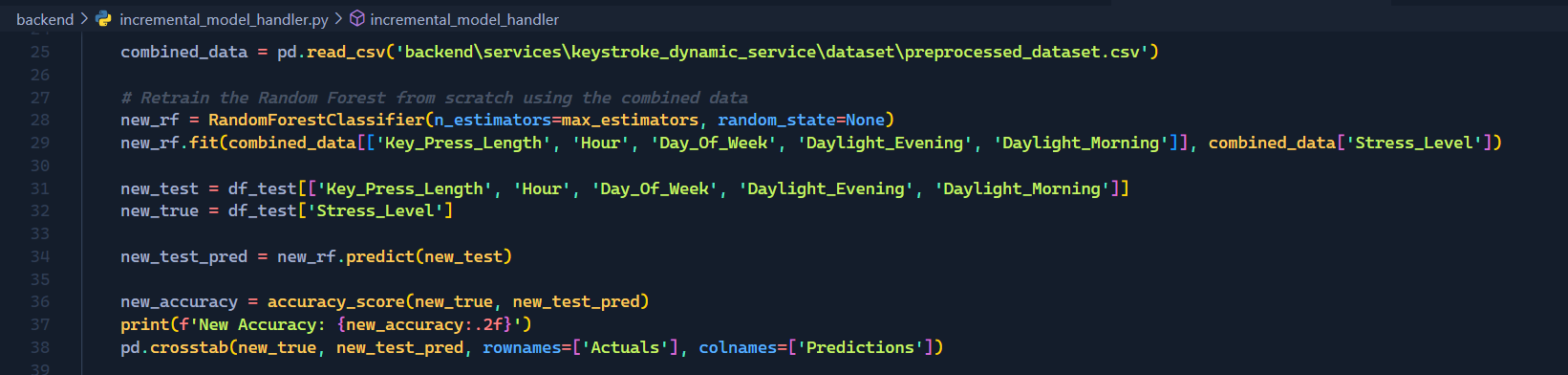


Figure 3. 33: Model Retraining and Testing New Accuracy

Once the model has been retrained the new model accuracy will be tested along with the test dataset and recorded.

However, the new model will not be accepted as the base model from the system unless the new accuracy is greater than or equal to the previous model accuracy.



Figure 3. 34: Compare old and new model accuracy

This function will be run every 24 hours. This will be triggered by a scheduled job which will run in a new thread evert 1440 minutes (24hours).

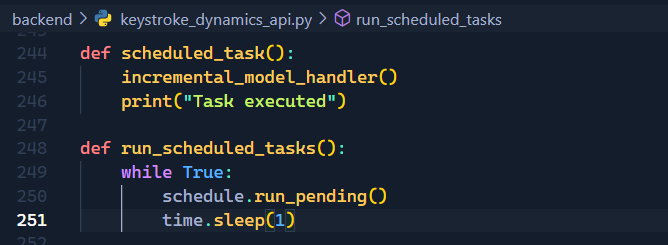


Figure 3. 35: Scheduler Function

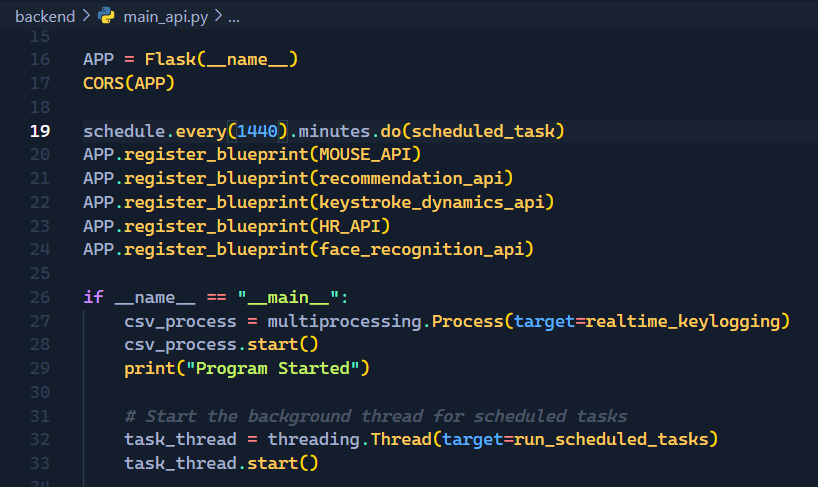


Figure 3. 36: Calling the function in 24 hours

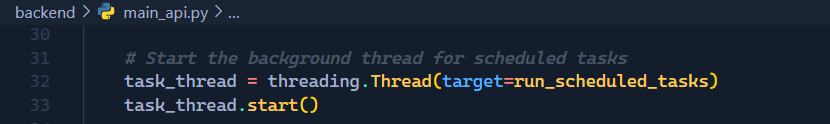


Figure 3. 37: Running the job in a new thread

This concludes the implementation process of the backend functionality of the main logic. This is the entire process that will be linked with the desktop applications’ keystroke dynamic based stress detection component.

**Desktop Application Integration**

The desktop application will be utilized by the user to represent the predictions gathered and also to interact with it and perform various activities with it.

The keystroke dynamic based stress detection component offers 2 options to the Desktop application. They are,

1. Realtime visualization of the detected stress level
2. On\_Demand keylogging with typing test

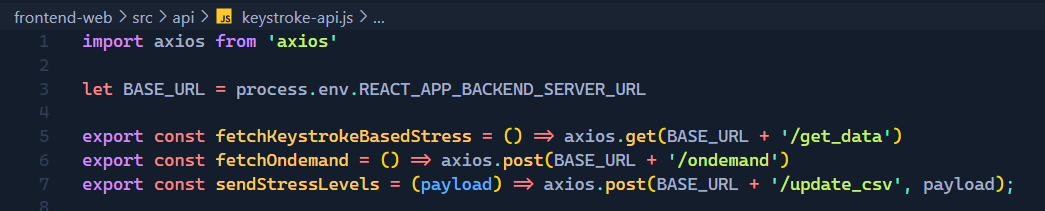


Figure 3. 38: Frontend APIs

The functionalities have been linked with API calls to the backend and each of them will be assisting in the integration process. The fetchKeystrokeBasedStress API will be called every 20 minutes from the frontend and it will send the stress level prediction generated from the backend. This data will be stored in a Redux state and will be represented in the dashboard. This API will continue to request new data every 20 minutes.

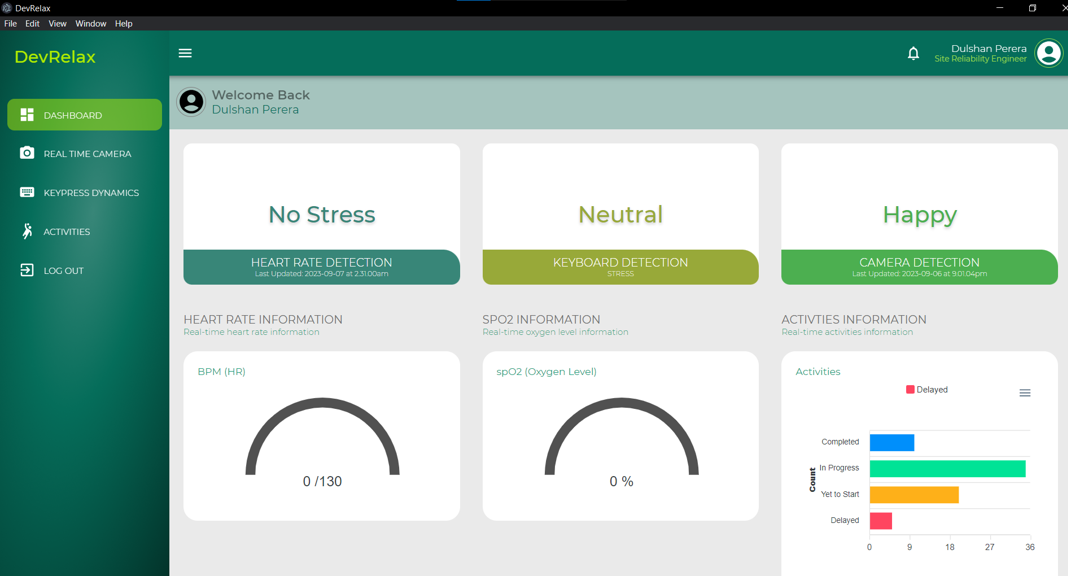


Figure 3. 39: Displaying gathered prediction data

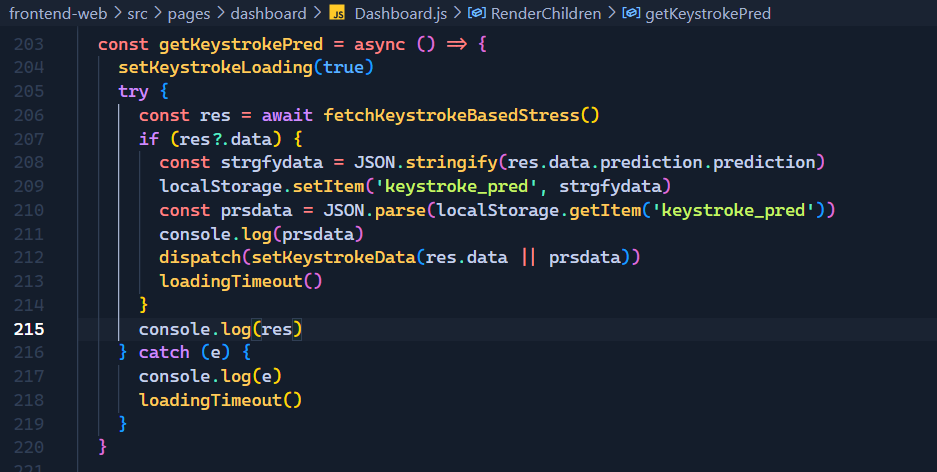


Figure 3. 40: Keystroke Dynamic based stress data retrieve function snippet

The getKeystrokePred function will acquire the keystroke-based stress level prediction and will save it in the Redux state and the local storage. This will be used to display the users current stress level in the dashboard. OnDemand keylogging will be utilized in this section where the user is performing the typing test provided in the Application. The call is triggered when the test is started. The test will run for one minute and the Typing Speed, Error Rate and the Accuracy will be acquired along with the Stress Level.

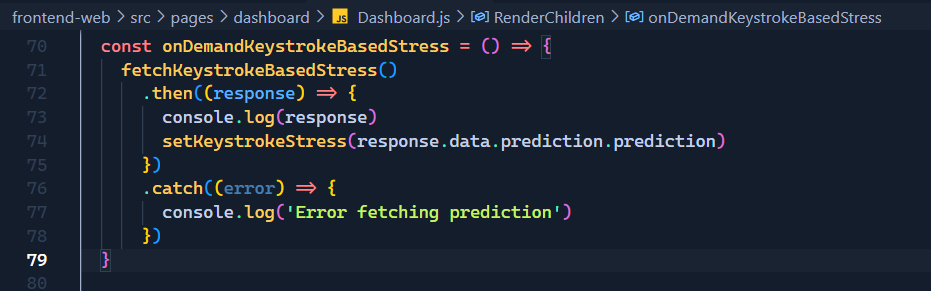


Figure 3. 41: OnDemand keystroke-based stress retrieve function

This will allow the user to test their stress levels when they require it without waiting for the 20-minute session to complete. However, this process will not stop the 20-minute session keylogging. It will continue along with this.

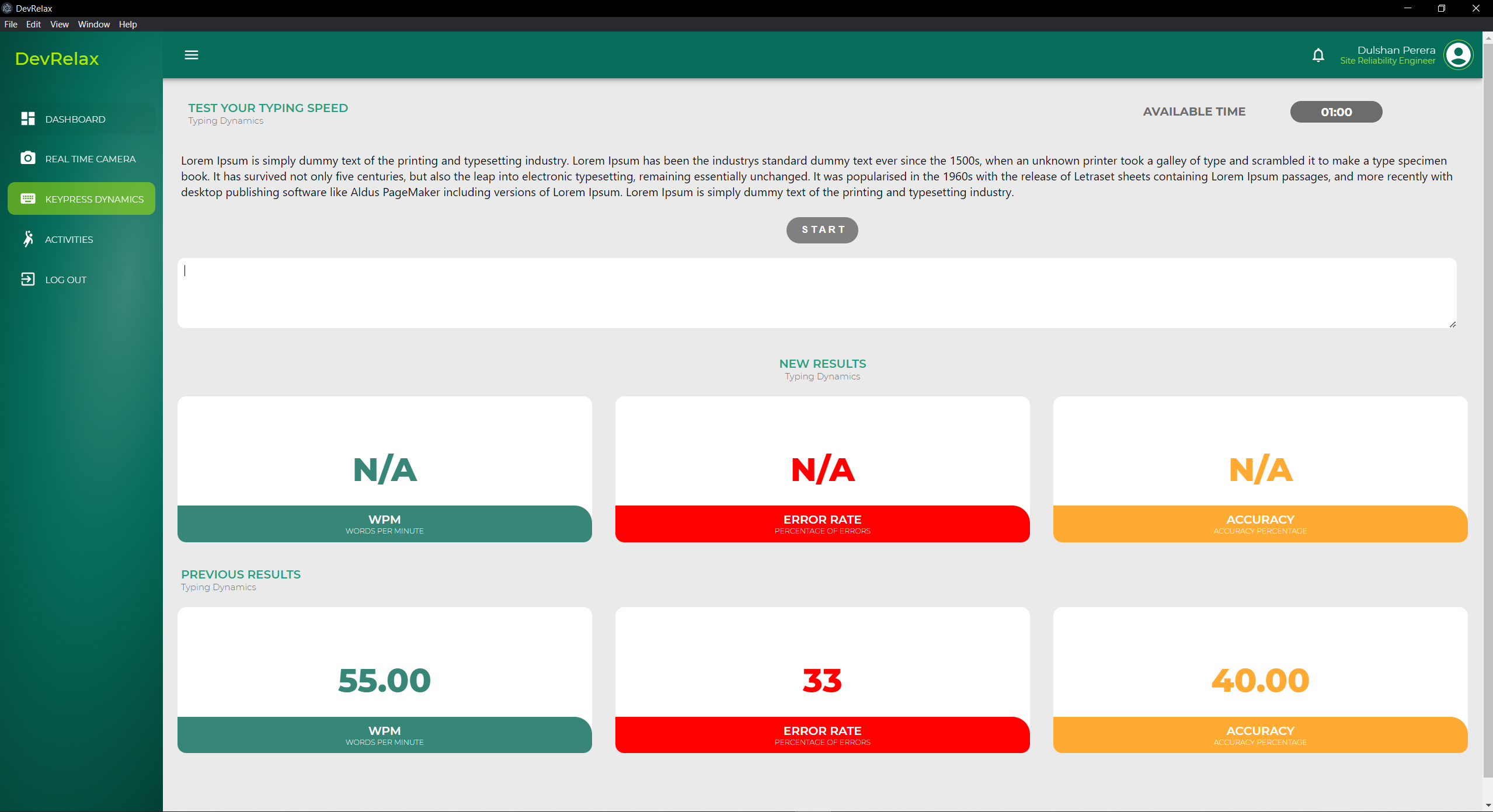


Figure 3. 42: Typing test dashboard

The aggregated stress value will be prepared according to the proper format before it is sent back to the backend for updating the CSV file.

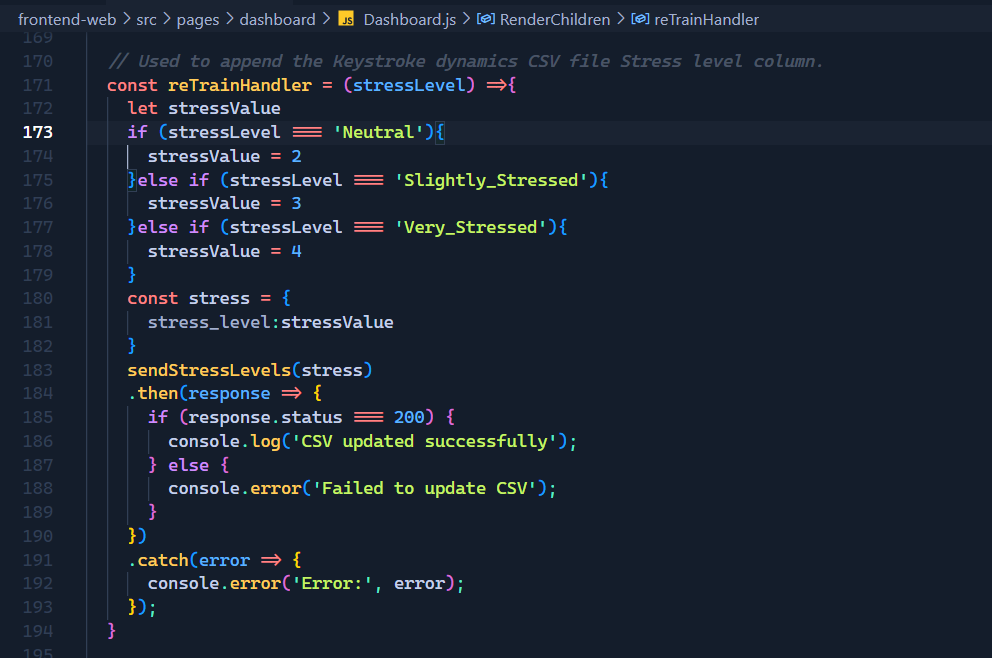


Figure 3. 43: Retrain handler code snippet

This explains the main points in the Desktop application integration of the keystroke dynamic based stress detection component.

This concludes the overall explanation and discussion on the implementation of the project and its methodology.

### Functional and Non-Functional Requirements

Below are the Functional and non-functional requirements of the component,

1. Functional Requirements

* That the program should be able to detect and filter the keypress latency related data.
* The program should be able to detect and filter the users typing speed variations.
* The program should be able to detect and filter the typing accuracy of the user during the typing session.
* The trained model should be able to predict the stress level of the user based on the above-mentioned inputs.
* The model should be able to learn from the new data.

1. Non-Functional Requirements

* Security
  + The keylogger should not save the key labels during the keystroke dynamic data logging phase.
* Performance
  + The component should quickly identify and detect the stress level of the user based on the keystroke related data.
* Usability
  + The user should not be troubled and with the program running in the background. It should allow the user to carry out their work smoothly.
* Availability
  + The program should be available throughout the user’s session and monitor the keystroke dynamics.

### Feasibility Study

Prior to the commencement of the requirement analysis, design, development, and testing phases, a feasibility study was undertaken to determine the product's schedule, technical, and financial feasibility

* Schedule Feasibility

Sprint-based solution implementation ensures a working product at sprint's end. Each phase must meet a deadline to deliver a high-quality output.

* Technical Feasibility

The solution depends on training models with machine learning algorithms to provide predictions, hence all team members must understand machine learning. As the application is intended to be constructed in React JS and Electron JS, an adequate knowledge of React JS and Electron JS is also required.

* Economic Feasibility

The solution must be affordable for all team members. Traveling and hosting must be affordable.

### Testing

Testing is an important step in a project. It ensures that the end product is functioning as expected. Therefore, the application is required to be tested in several levels.

1. Backend Level
   * Test API calls
   * Model Results
   * Model Predictions
2. Frontend Level
   * UI Tests
   * User Testing (Ongoing)

The backend of the application contains the Node JS backend as well as the FLASK server backend where the model predictions and keylogging functionalities are defined.

When discussing about backend level API calls all the backend API calls were tested using POSTMAN the tested API Endpoints were,

|  |  |
| --- | --- |
| API Endpoint | Request Type |
| <http://127.0.0.1:4567/predict> | POST |
| <http://127.0.0.1:4567/ondemand> | POST |
| <http://127.0.0.1:4567/ondemand-predict> | POST |
| <http://127.0.0.1:4567/get_data> | GET |
| <http://127.0.0.1:4567/update_csv> | POST |

Table 3. 3: API Tests

During the implementation phase a considerable amount of testing was incorporated during the model selection phase of the model training and development section. Therefore, what was remaining was to test the model with unseen data and during the application testing phase the models will be indirectly tested. The model was further tested by altering the datasets as well.

During minor application testing phases the model predictions were also tested on the stress level prediction variations. The variations include Neutral, Slightly\_Stressed and Very\_Stressed states.

Frontend testing was carried out every time a when backend component was integrated or when a frontend component was added. During these phases several components were tested,

|  |  |
| --- | --- |
| Test Case ID | Test Cases |
| T001 | Detects Neutral State |
| T002 | Detects Slightly\_Stressed State |
| T003 | Detects Very\_Stressed State |
| T004 | Detects Typing Speed |
| T005 | Detects Error Rate |
| T006 | Detects Accuracy |
| T007 | Resets the Dashboard |
| T008 | Starts the Timer |
| T009 | Produce Results when timer is 0 |
| T010 | Display Previous Results after the reset |

Table 3. 4: UI Test Cases

The tests have been completed and will be further discussed in the Results section of this report. However, the user testing phase is still ongoing therefore, only the testing method will be discussed as it is still awaiting results.

# RESULTS AND DISCUSSION

This section will systematically provide the important aspects of the test results, research findings, discussion and the limitations of the product with the appropriate diagrams and screen shots.

## **Results**

This section will discuss about the results that were gathered during the tests that were conducted during the testing phase. The tests were discussed in the testing section under the implementation and testing section.

As discussed earlier the backend API tests were carried out and the results are as follows,

http://127.0.0.1:4567/predict



Figure 4. 1: Predict Test Result

http://127.0.0.1:4567/ ondemand-predict



Figure 4. 2: Ondemand Predict Test Result

<http://127.0.0.1:4567/ondemand>

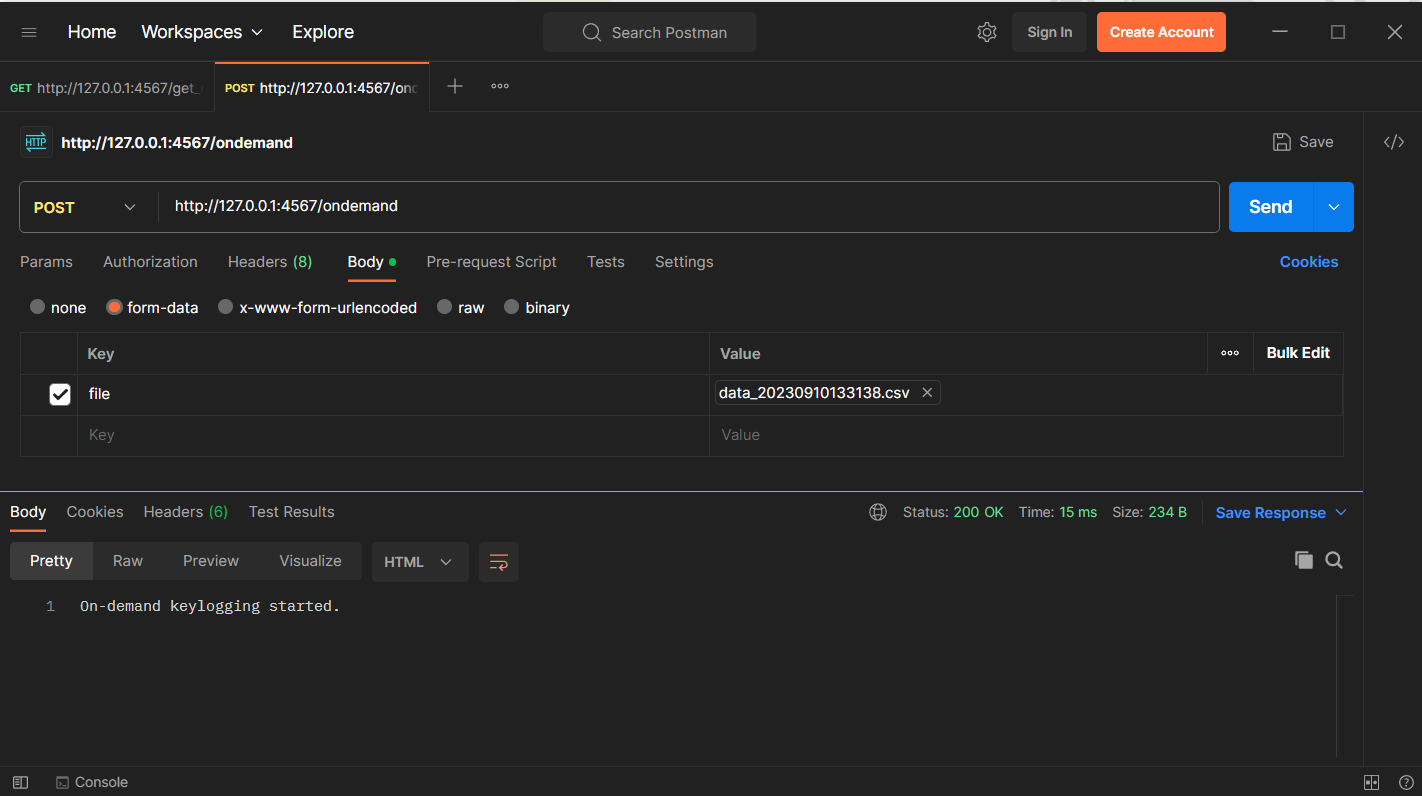


Figure 4. 3: Ondemand Test Result

http://127.0.0.1:4567/get\_data

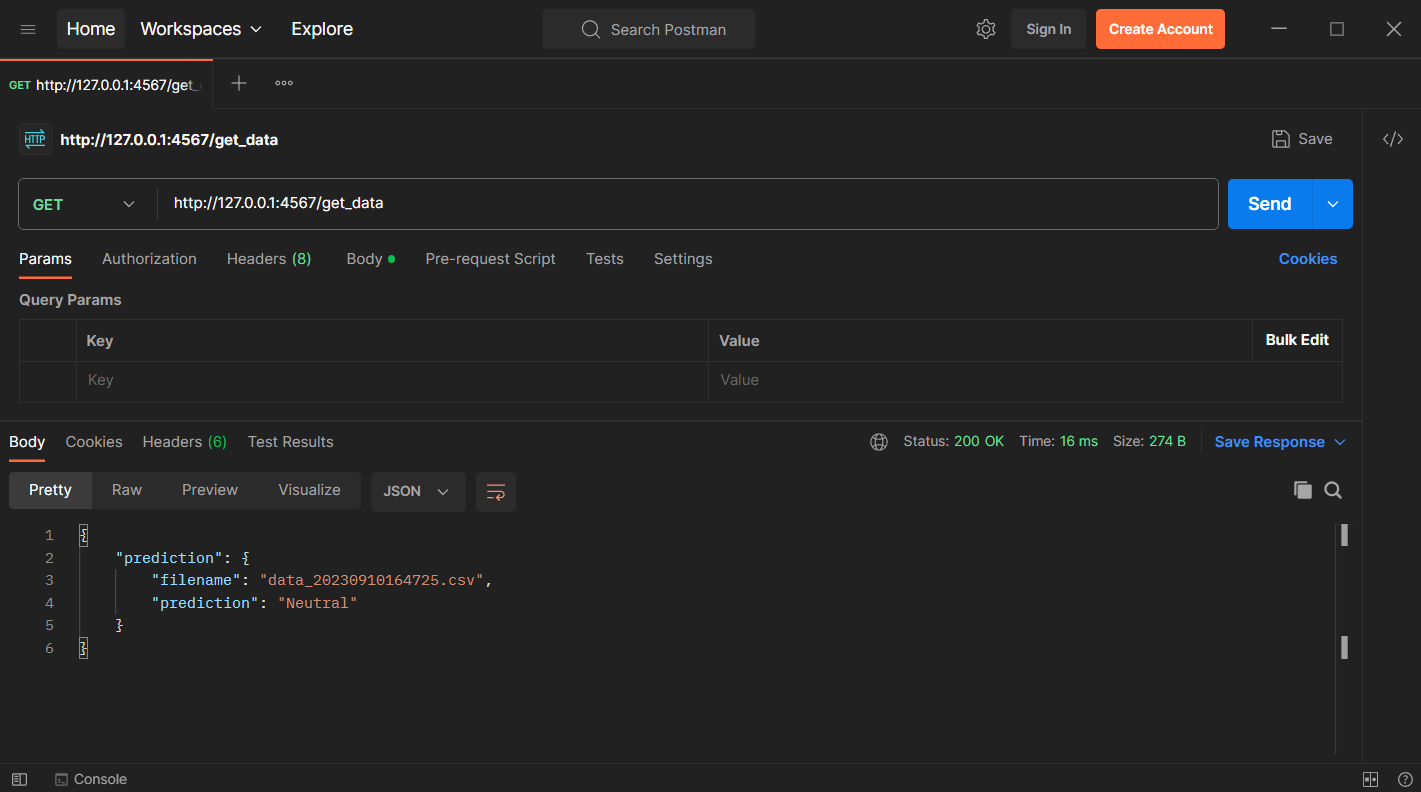


Figure 4. 4: get\_data Test Result

<http://127.0.0.1:4567/update_csv>

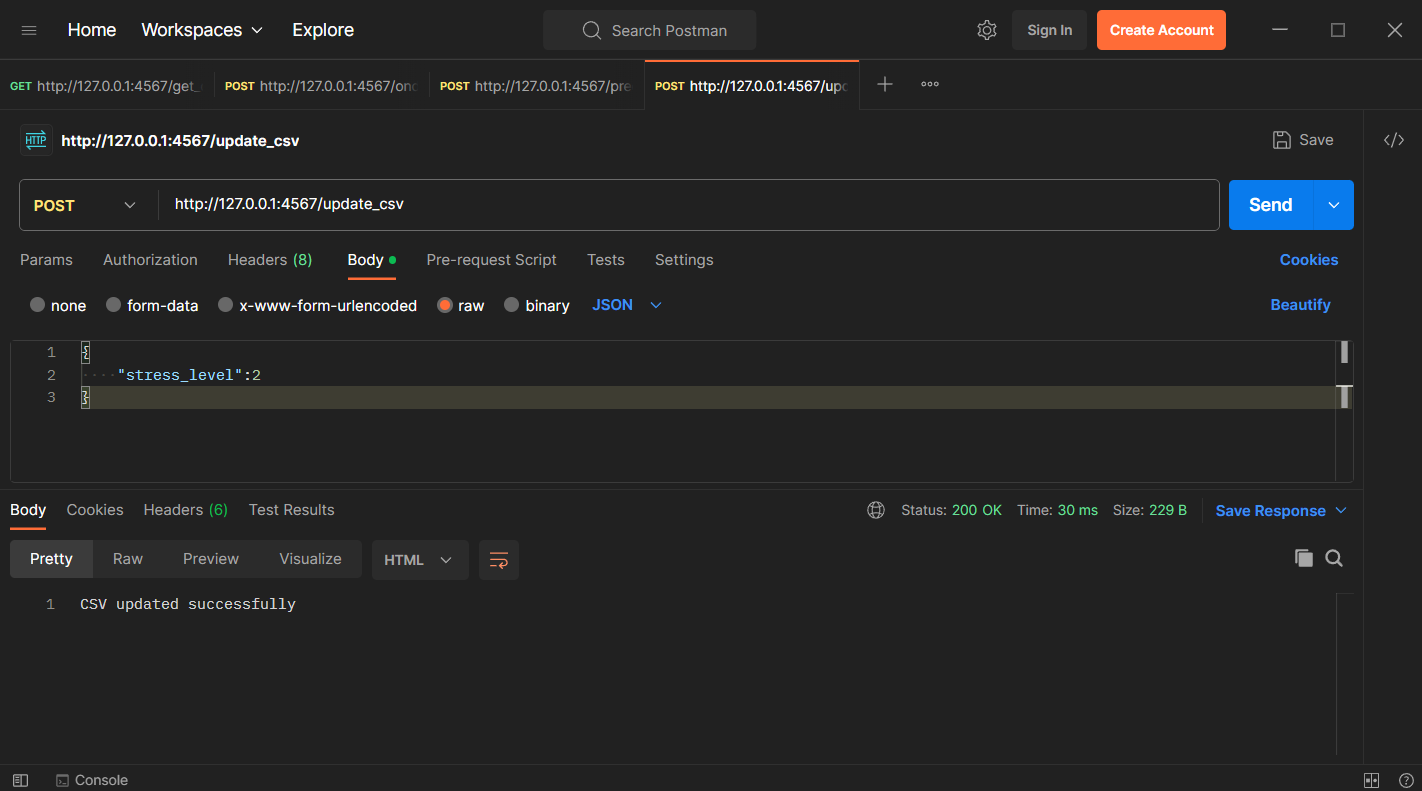


Figure 4. 5: update\_csv Test Result

|  |  |  |
| --- | --- | --- |
| API Endpoint | Request Type | Results |
| http://127.0.0.1:4567/predict | POST | Successful |
| <http://127.0.0.1:4567/ondemand-predict> | POST | Successful |
| <http://127.0.0.1:4567/ondemand> | POST | Successful |
| <http://127.0.0.1:4567/get_data> | GET | Successful |
| <http://127.0.0.1:4567/update_csv> | POST | Successful |

Table 4. 1: API Test Results

As per the results of the backend API calls all the APIs were functioning as expected and all the results were successful.

When considering the model testing. As explained in the above section the model has been tested a considerable amount of time using various approaches. The results and tests are discussed in detail in the Implementation section under Model Development. However, since the final selected model was Random forest the test results relating to the model will be discussed illustrated further.

As discussed earlier 2 datasets were used during the model selection phase,

1. Dataset 1: Had 12414 rows of keystroke dynamics related data.
2. Dataset 2: Had 14818 rows of keystroke dynamics related data.

Both datasets were used in the testing phase and the accuracy results are as follows,

**Dataset 1 Tests**

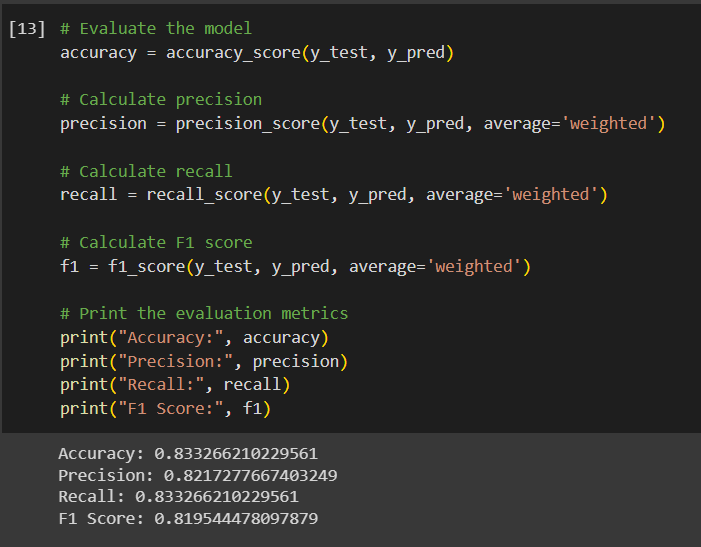


Figure 4. 6: Accuracy of the model without oversampling (DS-1)

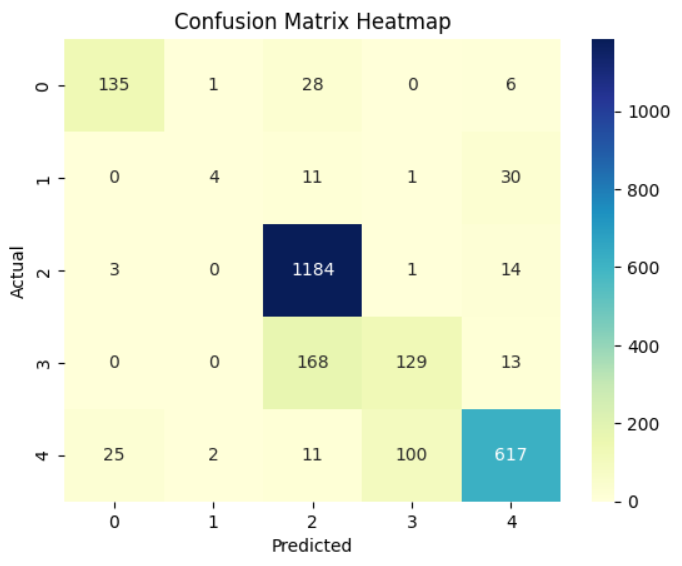


Figure 4. 7: Confusion Matrix Heatmap of the model without oversampling (DS-1)

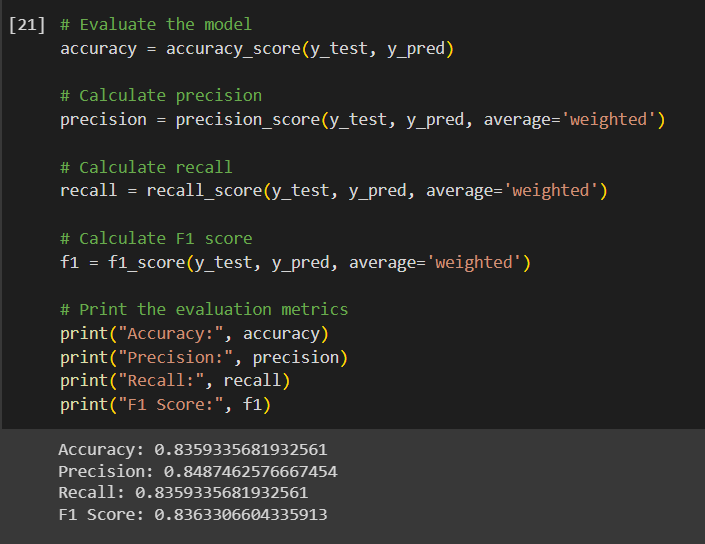


Figure 4. 8: Accuracy of the model after oversampling (DS-1)

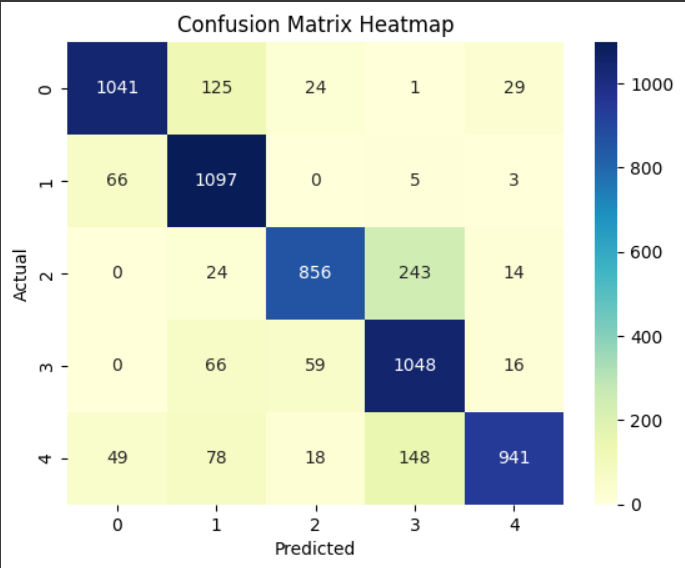


Figure 4. 9: Confusion Matrix Heatmap of the model after oversampling (DS-1)

**Dataset 2 Tests**

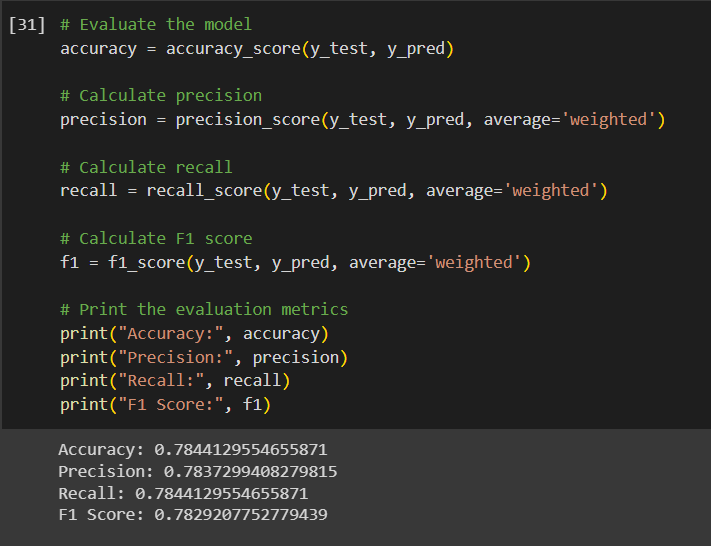


Figure 4. 10: Accuracy of the model without oversampling (DS-2)

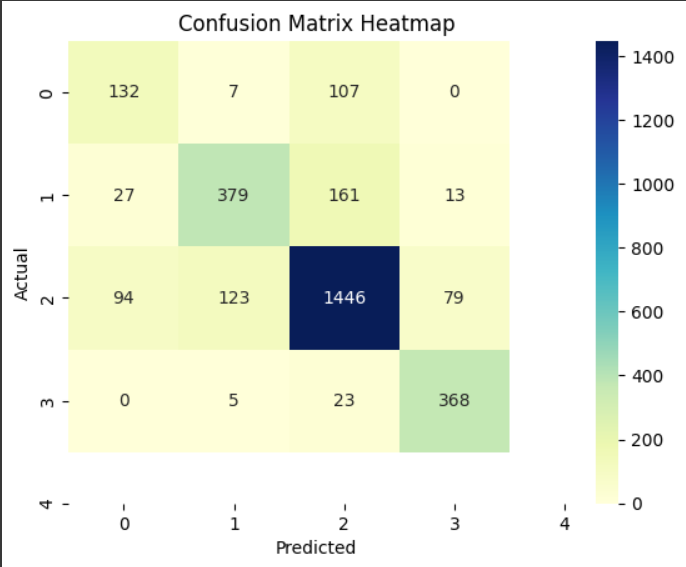


Figure 4. 11: Confusion Matrix Heatmap of the model without oversampling (DS-2)

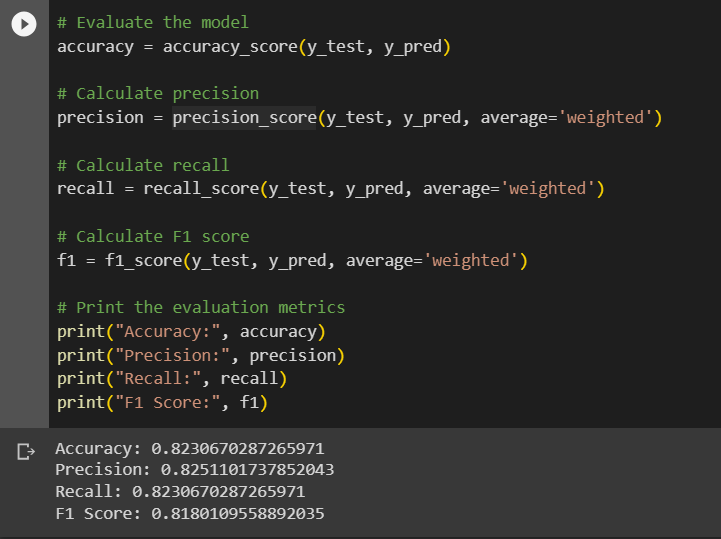


Figure 4. 12: Accuracy of the model after oversampling (DS-2)

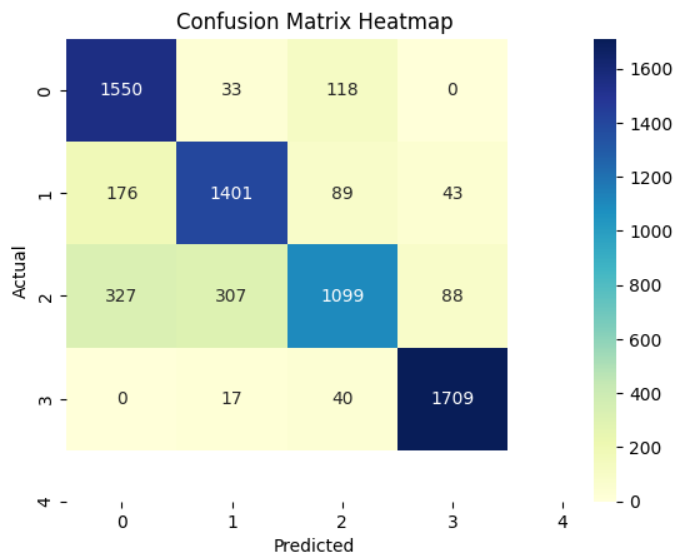


Figure 4. 13: Confusion Matrix Heatmap of the model after oversampling (DS-2)

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Final Accuracy** | **Final F1 Score** | **Selected/Not Selected** |
| Dataset 1 | 0.8359335681932561 | 0.8363306604335913 | Selected |
| Dataset 2 | 0.8230670287265971 | 0.8180109558892035 | Not Selected |

Table 4. 2: Model Test Final Results

The application was tested on various situations as explained in the above sections. However, the following test cases were created to ensure the functionality.

**T001: Detects Neutral State**

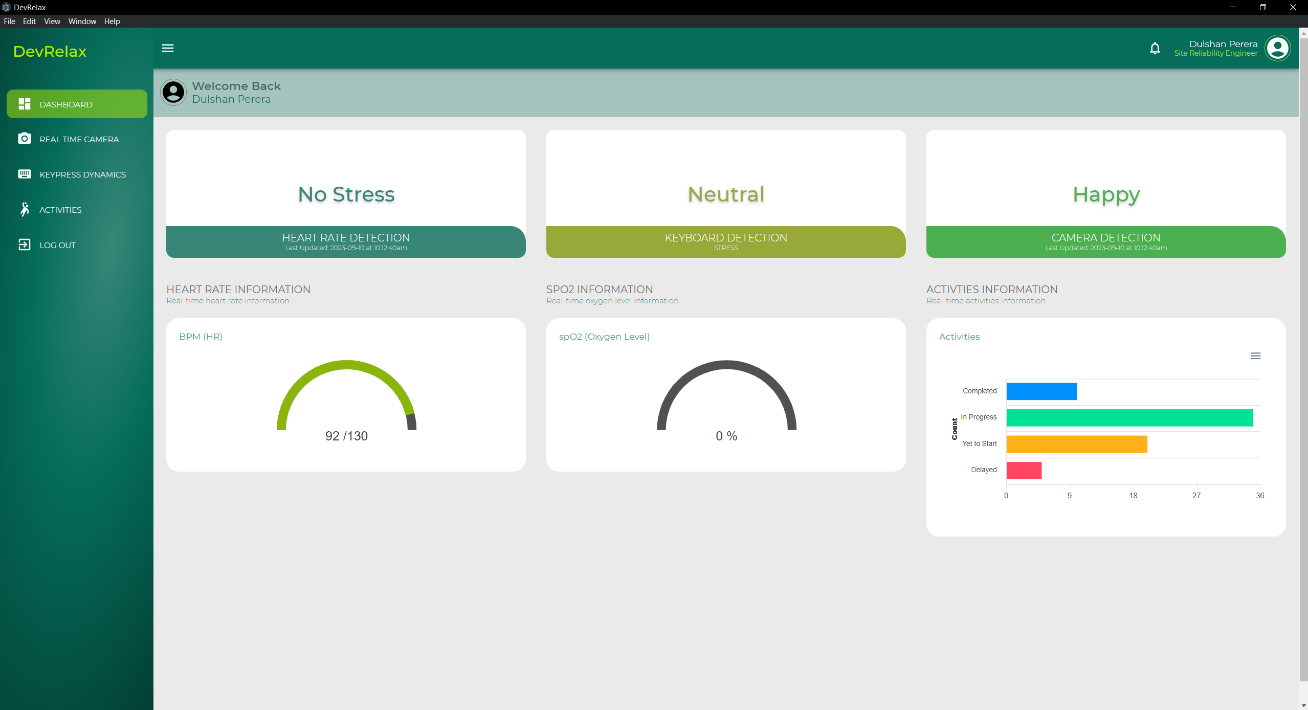


Figure 4. 14: T001 - Detects Neutral State Test

**T002: Detects Slightly\_Stressed State**

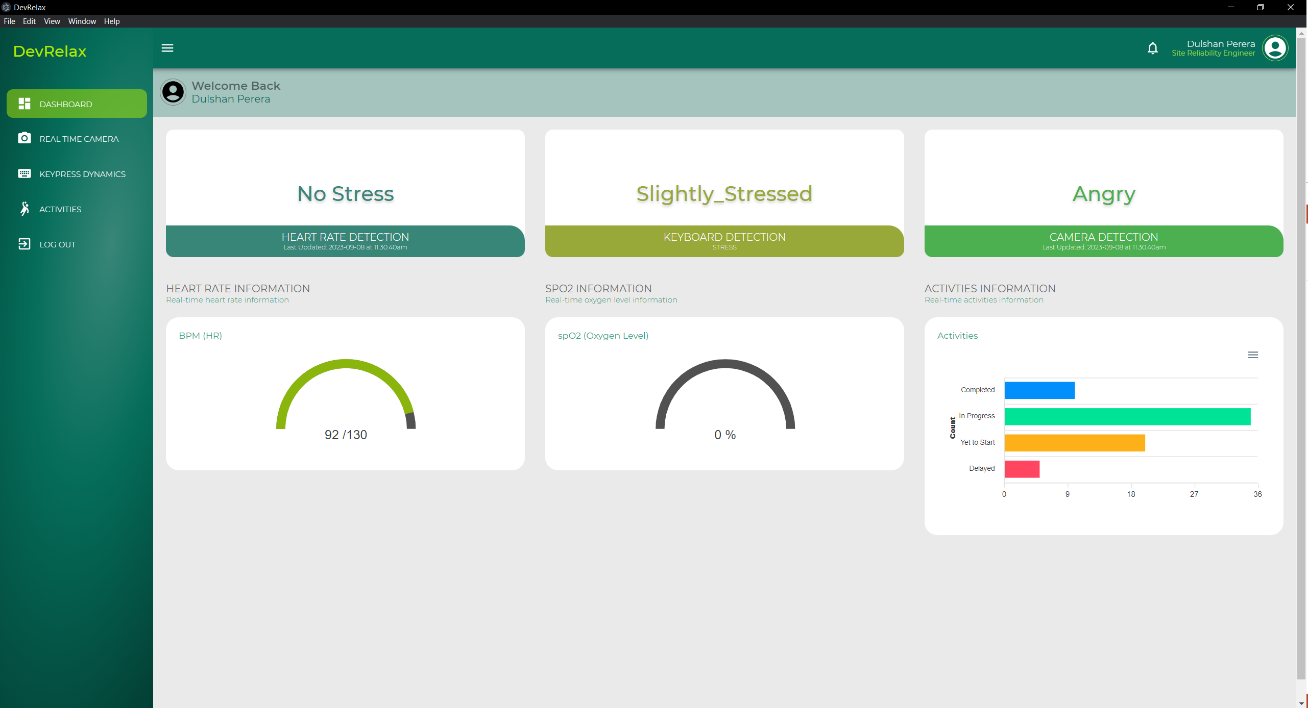
****

Figure 4. 15: T002 - Detects Slightly\_Stressed State Test

**T003: Detects Very\_Stressed State**

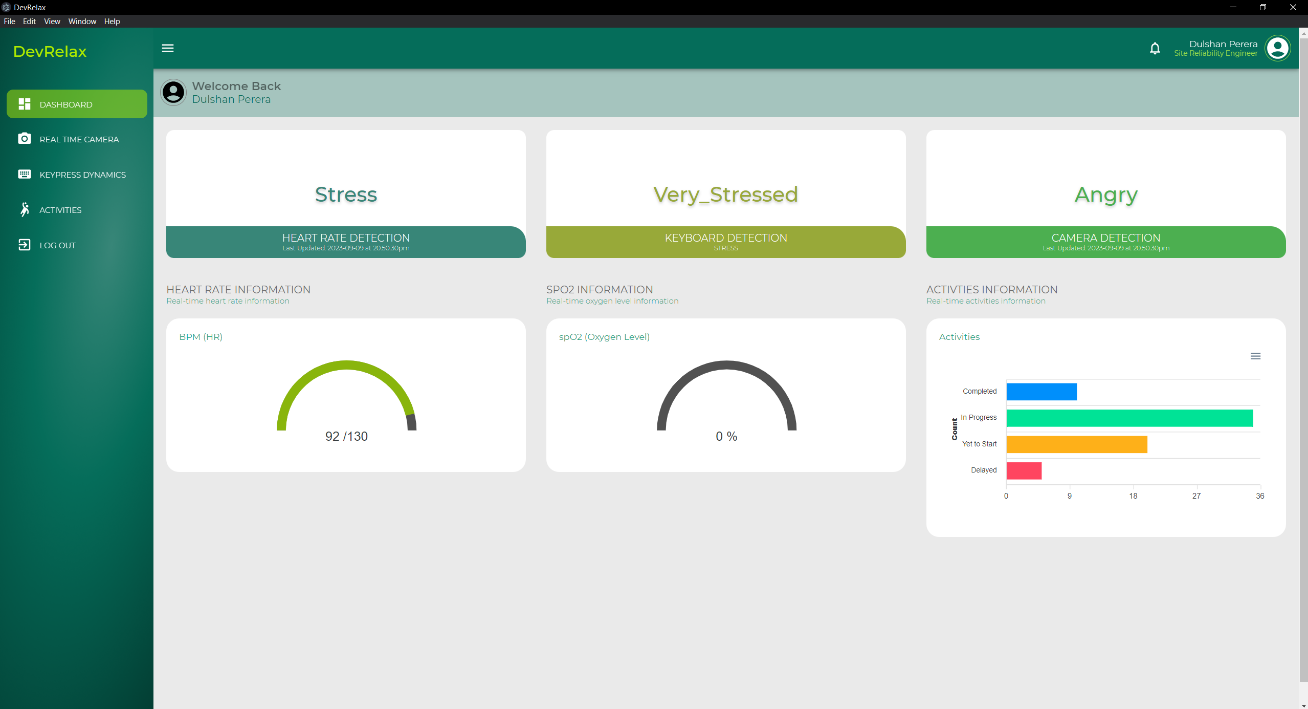


Figure 4. 16: T003 - Detects Very\_Stressed State Test

**T004: Detects Typing Speed, T005: Detects Error Rate, T006: Detects Accuracy, T009: Produce Results when timer is 0**

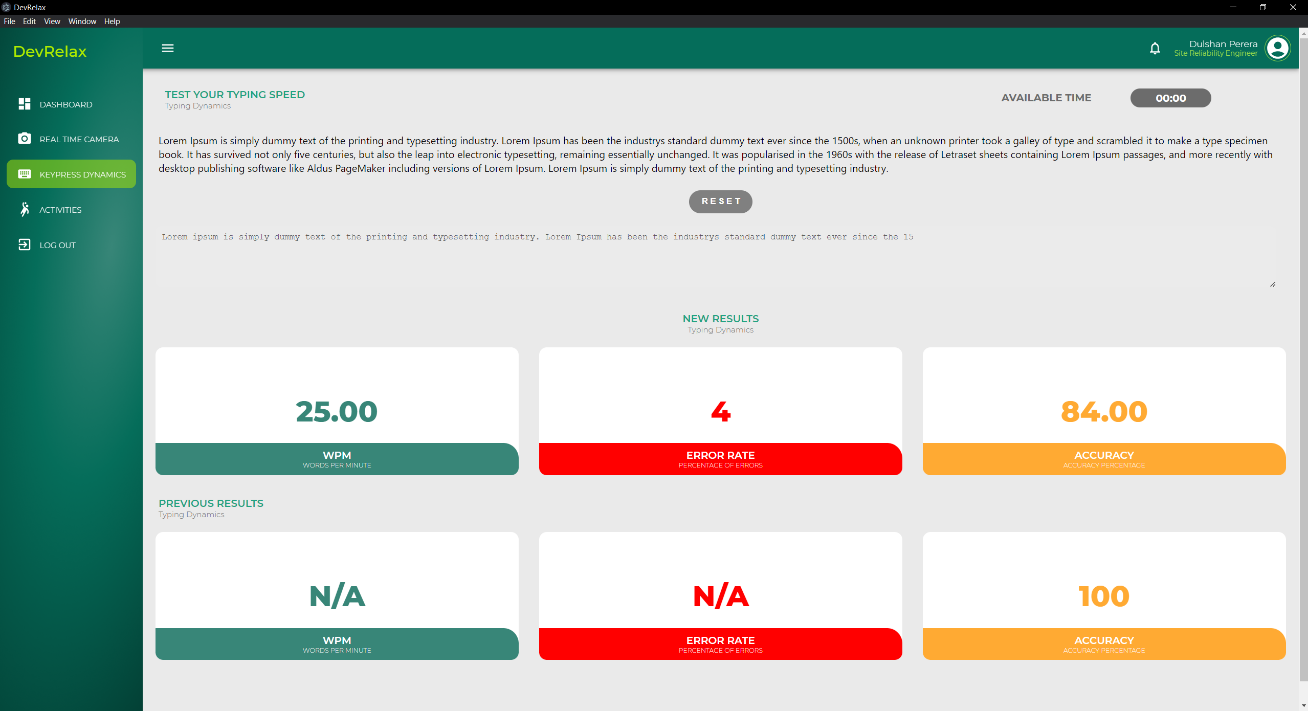
****

Figure 4. 17: T004, T005, T006, T009 - Test results

**T007: Reset the Test, T010: Display Previous results after the reset**

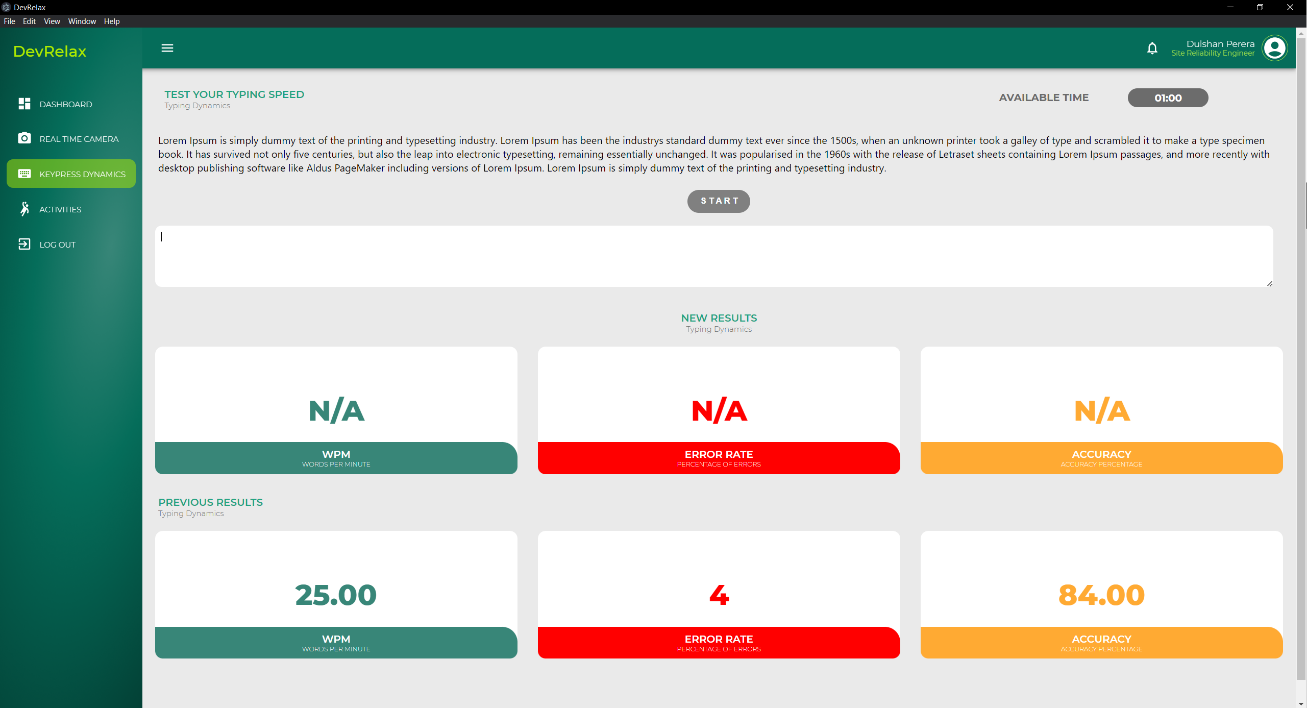
****

Figure 4. 18: T007, T010 - Test Results

**T008: Starts the Timer when the Test Starts**

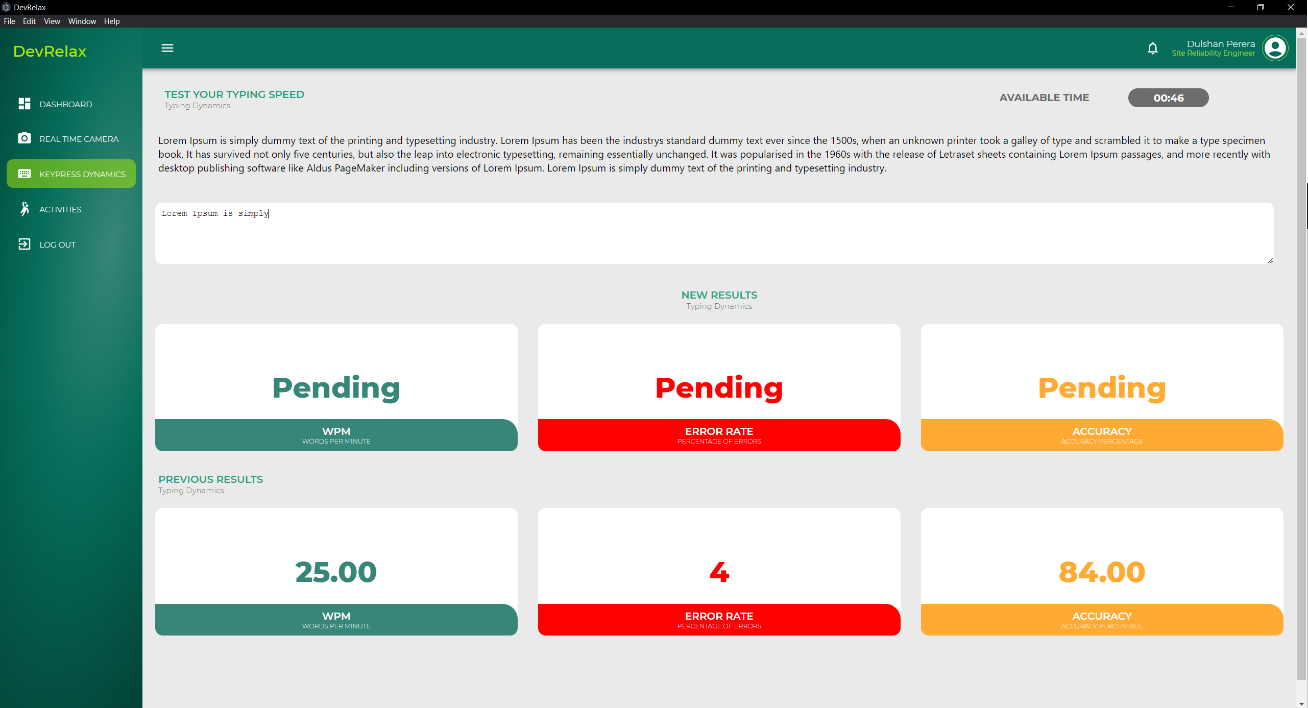
****

Figure 4. 19: T008 - Starts the timer test result

|  |  |  |
| --- | --- | --- |
| Test Case ID | Test Cases | Result |
| T001 | Detects Neutral State | Pass |
| T002 | Detects Slightly\_Stressed State | Pass |
| T003 | Detects Very\_Stressed State | Pass |
| T004 | Detects Typing Speed | Pass |
| T005 | Detects Error Rate | Pass |
| T006 | Detects Accuracy | Pass |
| T007 | Resets the Test | Pass |
| T008 | Starts the Timer | Pass |
| T009 | Produce Results when timer is 0 | Pass |
| T010 | Display Previous Results after the reset | Pass |

Table 4. 3: UI Test Results

The Desktop application still requires to be tested on the real environment under natural conditions. These tests are still ongoing. Therefore, unable to provide Test results for the ongoing tests.

## **Research Findings**

In the course of this research, several important findings were made. Firstly, it became evident that stress significantly impacts how users interact with their keyboards. This emphasizes the need to consider keystroke analysis in stress detection.

Additionally, it was observed that stress levels vary throughout the day, leading to distinct typing patterns in individuals experiencing stress. This highlights the connection between time of day and typing behavior.

A key breakthrough was the adoption of an incremental learning approach. Traditional models proved inadequate for capturing the unique nature of keystroke dynamics. The dynamic nature of typing patterns called for a model that could adapt continuously. By learning from new data, the model seamlessly adjusted to changes in users' typing behavior, ultimately improving the accuracy of keystroke analysis.

In conclusion, these findings confirm that it is possible to detect stress levels by analyzing keystroke dynamics.

## **Discussion**

The keystroke dynamic based stress detection component plays a huge part in the end product of this entire research. The component is able to detect stress levels of users without intervening with their natural course of work.

In contrast to traditional techniques that demand users to employ specialized keyboards or go through controlled stress induction, this approach leverages the natural use of keyboards in users' everyday tasks. This natural data collection method guarantees that users can carry on with their activities without being conscious of stress evaluation, thus minimizing the risk of influencing their behavior in a biased manner.

The component excels in gathering a wide range of keystroke dynamics, encompassing keypress length, press time, release time, accuracy, error rate, typing speed, and some other factors. This incremental approach results in a dataset that covers a wide range of user behaviors, ultimately refining stress predictions by considering various aspects that together indicate stress levels.

As detailed in the methodology section, the employed Random Forest-based machine learning model demonstrates impressive accuracy metrics, achieving a notable accuracy of 0.83593357 and an F1 score of 0.83633066. Notably, the model's adaptability is a key strength. Through regular retraining using freshly acquired data, the model refines its predictions and adapts to users' evolving stress patterns. This incremental learning mechanism ensures the model's continued relevance and effectiveness over time.

In the design of this component, user privacy and ethical considerations have been of paramount importance. By discreetly capturing keystroke dynamics, sensitive data or personal information is not involved, safeguarding user privacy. Additionally, the developed keylogger program automatically transforms any alphanumeric key label into a random symbol, further fortifying the dataset against potential information leaks. Furthermore, users experience no disruption in their usual keyboard usage during the assessment process.

The component seamlessly integrates into users' daily routines, facilitated by its background operation within a Flask server. The consistent acquisition of keystroke dynamics at 20-minute intervals guarantees uninterrupted data input without interfering with user activities. This adaptability to diverse environments, keyboards, and usage scenarios positions the component as a versatile tool for stress assessment.

In summation, the keystroke dynamic-based stress detection component presents an innovative and pragmatic approach to stress assessment. Its natural data collection nature, adaptive learning model, and smooth integration set it apart as a groundbreaking tool for comprehending and managing stress. By eliminating the need for specialized hardware or deliberate user engagement, this component marks a significant advancement in stress detection research. Overall making the component integration to the end product which is the Stress detection and relieving Desktop application a viable choice.

## **Limitations**

Just like every other research project this research component consists of its own limitations. The keystroke dynamic based stress detection component detects keystroke dynamics acquired from the keyboard. However, users do not always use the keyboard when they are working. They will be switching between the mouse and the keyboard, they may also be using mobile devices while working as well. Even though the switching between the mouse and the keyboard has been addressed by incorporating the external mouse with the HRV sensor in to the prediction output aggregation. The mobile device issue was not addressed.

This is considered as a limitation in the component. If this limitation needs to be addressed then it would require to acquire mobile application touch dynamics from the mobile phone and produce an aggregated output by combining both Keystroke Dynamics based stress prediction output as well as Mobile Touch Dynamics based stress prediction outputs.

This will be addressed as a future improvement to the component in the future releases of the application.

## **Summary of Each Students Contribution**

|  |  |  |
| --- | --- | --- |
| Registration Number | Name | Contributions |
| IT20037888 | Ranasinghe J. D. | Implemented the “Stress Detection via Facial Dynamics” component  Implemented the backend code for the component  Created the frontend application for the component  Created two different model architectures for the component  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |
| IT20037338 | Jayathilake S. M. D. A. R. | Implemented the “Stress Detection via HRV Sensors using Mouse” component  Implemented the backend code and the API routes for the component  Created four model architectures for the component  Created the frontend application for the component  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |
| IT20274702 | Bartholomeusz S. V. | Implemented the “Recommendation and Alleviation system” component  Implemented the backend code for the component  Created the frontend application for the component  Created a reinforcement learning model architecture for the component  Developed a list of activities that alleviate stress levels  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |
| IT20020262 | Perera M. S. D. | Implemented the “Stress Detection via Keyboard Dynamics” component  Implemented the backend code for the component  Created the frontend application for the component  Created the model architecture for the component and implemented incremental learning  Created a program for capturing keypress dynamics  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |

Table 4. 4: Summary of each student’s contribution

# CONCLUSION

In conclusion, this research paper introduces a pioneering method for detecting users' stress levels through the analysis of keystroke dynamics, leveraging the distinctiveness of individual typing patterns. The machine learning component, powered by the Random Forest algorithm, offers a seamless and ongoing assessment of stress levels without intruding on users' tasks. This stands in contrast to previous studies that relied on specialized keyboards, artificial stress induction, or bio sensors, highlighting the inherent value of analyzing natural typing behavior.

The adoption of incremental learning has proven instrumental, allowing the system to evolve alongside users' changing behaviors, thereby enhancing accuracy over time. This adaptability is a testament to the robustness of our approach and its potential for widespread application.

Deploying the machine learning model within a Flask server demonstrates the practicality of this approach in real-world scenarios. This not only provides a responsive platform for stress level prediction but also lays the foundation for future scalability and integration into diverse applications.

As a future improvement by addressing the limitation, this research aims to incorporate touch-screen key dynamics from mobile phones, further enriching the stress assessment process. By utilizing data from various sources, including keyboard and mobile phone interactions, an aggregated prediction could yield a more comprehensive understanding of users' stress levels. This, in turn, promises a more precise evaluation of the current user's stress level by comparing and synthesizing data gathered from both platforms.

In conclusion, this research not only introduces a novel perspective on stress prediction but also underscores the potential of machine learning and incremental learning techniques in creating adaptive and user-centric systems. As technology continues to advance, the insights gleaned from this study hold promise for further innovations in the realm of stress assessment and its management. This work serves as a foundation for future research and development in this critical area of well-being and performance.

# REFERENCES

|  |  |
| --- | --- |
| [1] | "3 types of stress and what you can do to fight them," [Online]. Available: https://www.betterup.com/blog/types-of-stress. . |
| [2] | J. Hernandez, P. P. A. R. and M. C. , "Under Pressure: Sensing Stress of Computer Users," Association of Computing Machinery, p. 10, 2014. |
| [3] | Paul H. Dietz, B. E. Jonathan Westhues and S. B. , "A Practical Pressure Sensitive Computer Keyboard," UIST, 2009. |
| [4] | T. Yamauchi and K. X. , "Reading Emotion From Mouse Cursor Motions: Affective Computing Approach," Cognitive Science, 2017. |
| [5] | L. M. Vizer, "Detecting Cognitive and Physical Stress Through Typing Behavior," 2009. |
| [6] | S.-H. LAU, "STRESS DETECTION FOR KEYSTROKE DYNAMICS," p. 232, 2018. |
| [7] | L. Hai-Rong, L. Z.-L. Y. W.-J. and J. D. , "Emotion recognition based on pressure sensor keyboards," IEEE, 2008. |
| [8] | H. Lv and W.-Y. W. , "Biologic verification based on pressure sensor keyboards and classifier fusion techniques," IEEE, 2006. |
| [9] | A. A. M. S. Y. M. Lim, "Detecting Emotional Stress during Typing Task with," p. 10, 2014. |
| [10] | D. Gunetti and C. P. , "Keystroke analysis of free text," ACM, 2005. |
| [11] | G. Hytry, "IT’s Stressful. Ask DevOps, They’ll Know: Stress in the IT Sector [2022 Survey]," 2022. [Online]. Available: https://spacelift.io/blog/are-it-jobs-stressful. |
| [12] | C. Weerasinghe, "Stress Detection by Keystroke,App & Mouse Changes," [Online]. Available: https://www.kaggle.com/datasets/chaminduweerasinghe/stress-detection-by-keystrokeapp-mouse-changes. |
| [13] | Z.-L. L. W.-J. Y. J. D. H.-R. LV, "EMOTION RECOGNITION BASED ON PRESSURE SENSOR KEYBOARDS,," 2008. |

# APPENDICES