FITASSIST: Stress Avoidance Through Posture Observation and Exercise Recommender

Ume-Kalsoom Waheed, Muhmmad Shahid, Atif Saeed, Riaz Ul Amin

Department of Computer Science Comsats University Islamabad, Lahore Campus
Department of Computer Science University of Okara, Pakistan

Abstract
Digital gadgets like desktop computers, laptops, and mobile phones have become an essential part of the modern life. Due to availability and ease of use of the high tech digital gadgets, everybody spends a lot of time on computers that can cause serious stress on a human body. Today most of the jobs involve staring at computer screens for hours. The computer specific jobs require people to continuously interact with the computer screen for at least 5 hours a day on average. Some offices take care of the computer ergonomics (science of designing a job, work space or equipment to fit the worker) but most of the employers and employees are unaware of this. The usage of these gadgets without proper body orientation can lead to eye, backbone, neck, and head stress. Long term use of these gadgets can cause side effects such as musculoskeletal problems, obesity, vision problems and stress disorders as well. This can be avoided by sitting in a correct posture and taking exercises of eyes and back at regular intervals. The objective of this study is to monitor the users posture, eye, and neck stress. The user will be suggested exercises and tips to practice at regular intervals. The proposed software will maintain a record of the users posture statistics that will be shown in the form of graphs. The results of application monitoring, blink rate, and posture statistics indicate that there is a considerable improvement and accuracy in the proposed model which also helps in improving users physical health and reducing mental stress.

Keywords:
Ergonomics, computer vision, stress, posture

1. Introduction

The Good posture helps to perform daily tasks with more energy, less strain and fatigue. It helps to reduce chances of obesity, diabetes, heart, breathing and digestion problems. In the present era, usage of computers has increased dynamically which is giving rise to computer vision and digital strain issues. The usage of computers has reduced the time people used to spend on physical exercising, playing etc. This has increased the risk of various health related problems that are being induced slowly to our living patterns. The motivation of this study is to minimize the health risks associated with computer usage. **According to slouching** causes the muscles to bear more burden to keep spine stabilized and protected. This results in changing the curvature of the spine and destruction of posture. We plan to develop an environment where the user can work and know when to relax. This will enable the user to work effectively and maintain a healthy lifestyle simultaneously. The eye problems due to computer usage fall under computer vision syndrome (CVS). While using computer, the eye muscles have to continuously focus and refocus. Moreover, screen adds flicker, glare, and contrast that require more effort from eye muscles. It makes the eyes dried, red and causes blurred or double vision. Major symptoms of CVS are shown in the Figure 1. The objective of this research is to help the gadgets users to reduce their mental stress and maintain and improve physical fitness through recommending exercises.

Figure 1. Computer Vision Syndrome

The poor neck and head posture leads to kyphotic posture that is excessive forward curvature of the spinal cord. This happens due to dropping shoulders and tensed back as shown in Figure 2.

There is a separate field of study named computer ergenomics that deals with the posture and orientation of body to reduce body strain. The user is prone to develop bad posture and eye strain issues with regular usage of computers. The application targets to remind the user to correct his posture whenever he is slouching and recommends exercises to overcome eye and back stress.

Manuscript received May 5, 2025
Manuscript revised May 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.5.18

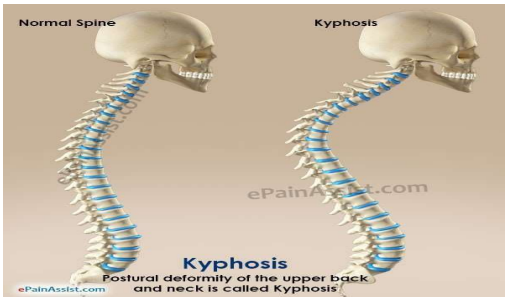

Figure 2. Team ePainAssist, Kyphosis: Treatment, Taping Technique, Causes, Symptom. 2013.

The rest of the paper is organized as follows: Section 2 showcases the related work. Methods and materials are detailed in Section 3. Section 4 presents the mathematical model. Experiments and results are discussed in section 5 followed by the evaluation in Section 6. Section 7 provides concluding remarks and suggests future work.

2. Related Work

Some considerable work carried out in computer ergonomics has been surveyed as under:

2.1 MacBreakZ

MacBreakZ [1] is a desktop application developed for Mac OS and works only for posture building and recovery from poor posture. It works by providing break reminders at intervals fixed by the user. In the break, application provides simple stretches and exercises. The application does not monitor if the user is actually following the exercises or sitting in correct posture. The application can be found on this website.

2.2 Posture Man Pat

Posture Man Pat[2] is another Mac OS application that uses accelerometer of the mobile phone or webcam of computer. It only gives alert when the Y-axis of webcam is below the defined target. It dims the screen and a small window is shown to user. The application only works to correct the neck position. There are wearables available too that get synchronized with mobile apps. These wearables can either use multiple cameras or accelerometers to check the posture. Cameras are expensive and are not applicable in daily life but accelerometers are comparatively smaller than cameras and are quite cost-effective. The official website of the software provides details of usage and tips.

2.3 LumoBack

LumoBack is a small wearable device that is worn near collarbone. It uses a single accelerometer embedded in belt to correct slouching posture. It slightly vibrates whenever you slouch. The data is transferred to a mobile application via Bluetooth that shows statistics and allows customization. The idea proposed in paper [1] uses 6 accelerometers placed on lower back, chest, right and left thighs, right and left leg. The sensors send data to mobile application, available on both playstore and itunes, via Bluetooth which performs calculations. It can distinguish between 4 postures i.e. sitting with bent knees, standing, lying face up and sitting with legs laid straight.

2.4 Zikto Walk

Zikto Walk [3] is another device that contains a gyroscope and accelerometer to monitor walking patterns. The mobile centric sensing as in [2] uses built-in sensor in mobile phones such as accelerometer, gyroscope etc. The compu- tations are performed using the capabilities of the mobile and no external storage is used. It has a drawback that a single accelerometer cannot clearly distinguish between standing and sitting. The mobile centric applications require mobile to be placed near the waist for best results. The paper [3] presents an idea of wearable vest that contains accelerometers at different places. The idea is to make elderly people wear the vest and enjoy life without the need of caretakers. The idea is more efficient as compared to tele-surveillance and continuous monitoring by a person. Similarly, the solution presented in [4] uses a pressure sensing mattress. The mattress is used to check whether the patient is lying in correct position and is comfortable or not. The mattress sends the data to computer workstation which evaluates the results. It alerts the related hospital staff whenever the patient's posture is incorrect. These solutions widely depend upon the external hardware resources for computations and storage. Its major disadvantage is scarce availability of the wearables.

2.5 BeUpright

BeUpright [5] is mobile application to benefit users by targeting the precise sitting posture. BeUpright senses two kinds of bad sitting postures which are leaning backward and leaning forward. In order to sense the position of a user BeUpright works on a slight sensor Bluetooth wireless. The sensor is worn on the user's shirt with the help of a magnet. When the connection is built in between the sensor and the application, BeUpright starts monitoring. When the user has bad position for more than 20 seconds, BeUpright gives an alert. After receiving the alert if the user does not alter the posture within 10 seconds, BeUpright locks the mobile of the user's helper.

2.6 A Surveillance System Designed for the Correction of Sitting Posture in Writing

A proper sitting position for writing that can be described with one of the most significant aspects is the distance D calculated from head to desk-plane. The systems proposed earlier only make the user attentive when his/her head is closer to the pen whereas the proposed system alerts the user when his head is far-off from the pen as well. This development can warn the user to abstain from sitting in the improper positions such as bending over the desk, head tilt etc. The proposed system consequently digitizes the production voltage of the sensor and inputs the data to a programmable regulator to check whether the D is in the safe series or not. Subsequently, the proposed system is composed of an infrared signal sensor, a programmable controller, an alert device, and a touch-switch [6].

2.7 Recognizing Posture in Pictures with Successive Convexification and Linear Programming

The approach presented in [7] made a posture detection technique grounded on local image structures and successive convexification image matching. Image matching built on successive convexification functions is contrary to the earlier approaches for example graph cut, belief propagation, iterative conditional modes (ICM), relaxation labeling, and additional convex programming-based optimization systems. The proposed system denotes target points for each pattern point with a small basis set. Successive convexification slowly makes the trust region for each template site smaller and alters the original difficult problem into an arrangement of easy and simpler convex programs. This hurries up the searching process, creating the technique suitable for large-scale matching and posture-recognition difficulties.

2.8 Technologies for Monitoring Students Body Posture and Physiological Parameters during Learning

Real Automóvil Club de España (RACE) [8] has carried out a research according to which 30% of traffic mishaps occur due to drivers' exhaustion. It is possible to recognize driver's fatigue from visual structures (actions, eye sight) or physical constraints (heartbeat, brain actions and pulse). If we can measure the driver's situation each second and practice that statistics in the car embedded system, then it is possible to build a driver assistant system that will decrease the possibility and number of road accidents. The symbols of tired look include degree of eyelid opening, person's head place and blinking frequency of eyes. This system comprises two core devices one of which converts information from analog to digital form. The other one gathers and modifies the information received, and directs it to the processing center. The central part of the system is the capability to evaluate and manage the symbols of driver tiredness in the real time. The system interferes in the form of alarm or signal and makes the driver attentive. Previously technologies exist for noticing persons through their features (hands, foots, head, eyes). These are some options for detecting human identity:

- Approaches for identifying persons using video.
- Using keyboard or mouse for tracing information.
- Sensors embedded on the body or seat.

Context Aware Posture Recognition in Offices (CAPRIO) is a system which builds a smart monitoring system that is for worker postures in office location. They used sensors which are embedded in the seats of workers. They uses pressure sensors and set them in back pane. With the help of these sensors, it can be clear whether the worker is sitting in correct posture or not. Research is being carried out to develop more effective systems.

2.9 Logtale

LogTale [4] is a desktop application that uses webcam to keep log of user's posture and issue notifications regarding posture. During the initialization process, the app monitors user for 1 to 2 minutes and checks his best, worst and comfort position. After that, the posture is evaluated by comparing vertical position of face **relative to camera coordinates**. Notifications are generated to notify user about his/her posture and a graph is used to show posture quality over a period of time. The application uses Pomodoro [5] technique to divide work session into an interval of 25 minutes. After every work session, a break should be taken. The application generates chart showing how much time user has been working and resting. We propose to use the idea of face detection as used in "slouch cam" application. The application defines a target zone and then detects user face. If the face goes below the target, an alert is given. We plan to add features in the same idea as it only works for slouch detection. Our aim is develop software that would perform the following tasks:

- The software will monitor user's posture.
- It will check user's dizziness and eye strain by tracking eye movement and eye blinking rate.
- It will give reminders for water intake.
- It will generate charts for average time spent on work and exercise that will help user to assess working patterns.
- It will check user's distance from computer screen and prompt whenever distance is too short.
- The software will prompt for exercises and will provide tips and facts when required. The study will yield a desktop based application for monitoring user's posture. The application will be packed with tools and technologies to help correct posture and sitting habits. It will reduce the risk of computer related injuries.

3. Methods and Materials

3.3 Hardware Configuration

× In order to run this software swiftly, it is necessary to use upgraded hardware configured in the system. In our experiment "Posture Monitor", we configure 4 GB of RAM to let software run smoothly along with other applications. Disk capacity in system should have at least 100 MB of free space for installation and configuration of software. There must be a working webcam either external or internal with at least 2 MP resolution. The application on a separate module has also been implemented that entails Raspberry pi 3, camera, screen and power bank. Raspberry pi is composed of RAM, CPU and memory card. The model of RAM is LPDDR2 (900 MHz) and size is 1 GB. CPU's model is 4 ARM Cortex-A53, 1.2 GHz. Memory card's model is MB-MP08D and storage capacity is 8 GB [6]. The model of camera is Pi camera module and its resolution is 5MP. Power Bank's model is A200, battery capacity is 7000 mah and output voltage is 2.1 A.

3.2 Sequence flow

User starts the application from system tray. The application loads the training data of existing users, if any. Next, the face recognizer is trained using existing data. Then webcam is used to capture user's image. The image is cropped and converted to gray scale as the recognizer works on gray scaled images. The captured image is compared with existing training data using openCV's LBPH recognizer. It returns label of the user it resembles and a confidence value. If the user is recognized as a new user, his/her information and settings are loaded and 10 sample images are taken using camera. Furthermore, a database file is created for keeping record. User's monitoring starts this way. If some users are recognized as the existing users, their blink rates are observed. Their computer usage is calculated by evaluat- ing time for which the users are available or unavailable. the posture is monitored by taking measures x, y at start when samples are taken. Then current values from video streaming are compared with those saved at start. Activity monitoring is done using activity monitoring application. It contains details of the applications and their usage time. It calculates the user' stress and busy hours in a day. After calculations, the results are loaded into database files to generate alerts or notifications suggesting rest or exercises on the basis of the stress types. Figure 3 shows the flow of application.

3.3 Face detection

Face detection is done using OpenCV's recognizer "LBPH" Towards data Science (2017). Face Recognition: Understanding LBPH Algorithm [online].Website

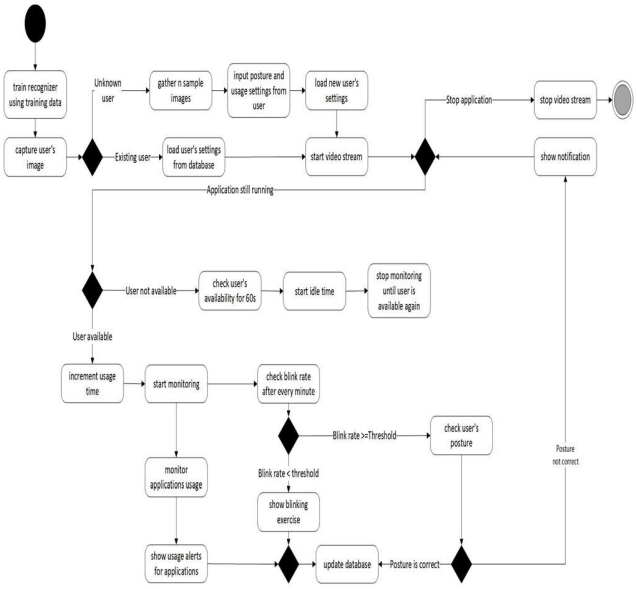

Figure 3. Sequence flow diagram

https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b [accessed 2018]. LBP Local Binary Pattern is a simple as well as a very effectual texture operator. It tags the pixels of an image by thresholding the neighborhood of each pixel. It takes the result as a binary number. LBPH uses 4 parameters.

Radius: the radius forms the spherical local binary pattern. It symbolizes the radius around the central pixel.

Neighbors: circular local binary pattern is produced using the number of sample points. These sample points are neighbors. If you add more sample points, the computational cost becomes higher.

Grid X: it is the count of cells horizontally. In order to have better grid and greater dimensionality of the subsequent feature, more cells should be added.

Grid Y: it is the count of cells vertically. In order to have better grid and greater dimensionality of the subsequent feature, more cells should be added. The Figure 4 below shows the procedure:

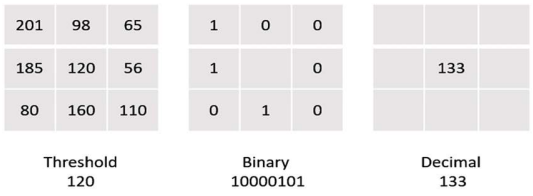

Figure 4. Working of LBPH recognizer

3.4 Blink Rate

Blink rate is evaluated using Dlib[7]. Dlib is a fresh C++ toolkit. It consists of machine learning procedures and tools for developing multifaceted software. It is used to solve real world problems for both industry and academia extensively. It provides 68 facial landmark points. The coordinates of these features can be used to calculate blinks. In order to detect blinks we use EAR (Eye Aspect Ratio) method as illustrated in [9]. Using Dlib we are able to get 6 points for each eye labeled in Figure 5.

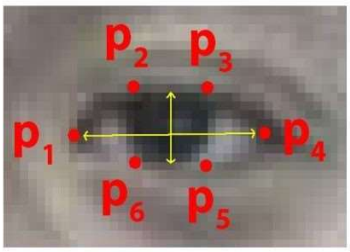

Figure 5. Eye blink detection with OpenCV, Python, and Dlib – PyImage-Search

The coordinates of these points are used to calculate EAR using the formula 1:

$$EAR = (|P_2 - P_6| + |P_3 - P_5|) / (2 \times |P_1 - P_4|) \quad (1)$$

The value of this ratio is calculated by dividing the vertical distance between the points with width of the eye. When the eye is open, the ratio yields some non-zero value. But when a blink occurs, the ratio drops to zero as the numerator i.e. the height of eye becomes zero[8].

3.5 Posture detection

Posture Detection is done using OpenCV's face detection module. During face detection, it forms a square box around users' face. The coordinates of that box are used to detect posture. We extract four coordinates from the bounding box: width, height and x, y coordinates of the top left corner. Another parameter called range is also used to specify the sensitivity of the calculations. Following formula is used to determine the head posture of user. If the new x- coordinate is less than the saved one, it means the user is moving to the left of the given coordinates:

$$left = x_{old} - x_{new} \quad (2)$$

If the new x-coordinate is greater than the saved x plus the width and range, the user is to the right of the given posture:

$$right = x_{new} - (x_{old} + width) \quad (3)$$

If the new y-coordinate is greater than the saved one, then the user is slouching that is named as bottom parameter:

$$bottom = y_{new} - y_{old} \quad (4)$$

If the new width value is greater than the saved value, the user is too close to the screen. this is calculated as:

$$close = width_{new} - width_{old} \quad (5)$$

If the new width value is less than the saved one, then user is far from the screen which is calculated as:

$$far = width_{old} - width_{new} \quad (6)$$

Based on these calculations if more than one parameter is positive then we find the largest from them and show its notification to the user.

3.6 Activity Monitoring

Activity monitoring is done using third party application "Application Logger" available as an open source project[9]. It generates the text file of logs containing the name of application and the time it was used. We extract the names of the applications and their duration using python code and use this data to check user's stress level. If user is switching between five or more applications in very less time, it means user is in stress and not able to do work. Otherwise, we can check which application is being used for how long time.

3.7 Statistics

Statistics are displayed in form of graphs. Blink rate, correct posture duration, bad posture duration, usage time, idle time of user and application usage patterns are the quantitative data being used for statistics. This data is manipulated to give different useful results in the form of graphs. Graphs are forms using python library "matplotlib". Matplotlib is a Python's two dimensional library used for plotting quality figures. Matplotlib helps to make difficult things possible by just few lines of codes. It allows you to produce plots, histograms, error charts, power spectra, bar charts, scatterplots, etc.

3.8 Notifications

Notifications are displayed using python library "plyer". Plyer is a platform-independent API **which most of the features are commonly found on many platforms**. Plyer does not try to reinvent the wheel but call for external libraries, depending on the current platform to implement the API in the simplest way. The application offers the user two ways to receive notification i.e. either sound or balloon notifications.

3.9 Sounds

Sounds are added using python library "pyttsx". Pyttsx is a Python suite which supports text-to-speech engines on Windows, Mac OS X and Linux. It converts the given text to speech.

3.10 Exercises

The exercises are based on the data collected from the user monitoring. The application recommends the exercise whenever blink rate is below the threshold. There are exercises for posture and stress. The application also displays a screen whenever the user exceeds the set usage limit. The screen can only be closed when user spends some time away from the computer.

3.11 SDLC Model

The Software Development Life Cycle Models have been used for the development of this application. We are going to use Iterative Model. The iterative model is a certain execution of a software development life cycle that emphasizes on an early, basic implementation which then gradually becomes more complex and broader features are added gradually and the final system is complete. Due to uncertain requirements of what would be easy for the users and chances of changes in requirements, the iterative model is preferred. Development of interfaces and implementation of simple algorithms were done at first. Then, gradually the complex functionalities were added to the application. This model helps us to incrementally improve the complexity of the system. Once the module is working and is complete, more features can be added to it.

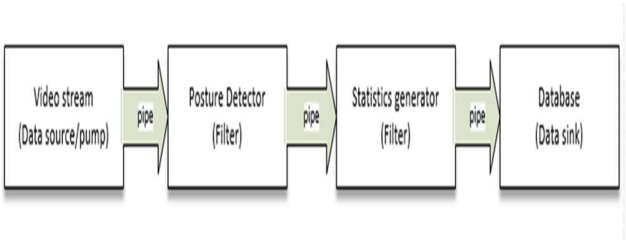

Figure 6. System Architecture (Pipe and Filter).

In pipe and filter style as shown in Figure 6, each filter has a set of inputs and outputs. The connectors are pipes and input data. The pipe and filter architecture is the most widely used architecture for computer vision applications. The video which take output of one filter and pass it as input to the next filter. Sequential flow of information is followed in this architecture. Source and sink filters produce only output stream acts as data source as it produces data that is passed to filter. The posture detector filters the data stream and passes the results to the next filter i.e. statistics generator. This filter passes its outputs to sink as input. The data sink stores the overall results generated by the system.

4. Mathematical model

For forming basic formula for calculation of stress used by FitAssist, we pick each currently running application and then we find out the following details repeatedly for each application :

- Extract the minutes that it was being used in and find blink rate. Compare blink rate with normal rate.
- Normalize by dividing with 100
- Determine the application type
- Determine the posture for that minute
- Multiply pose with app type
- Sum pose and blink rate
- Repeat for the duration of application usage

The following parameters are used:

- Usage duration (u): the total time user used the compute r
- Total apps used: total apps: the number of applications used during time u
- J= the start time for application i
- Application usage duration (a): the total minutes user used a single application
- Blink rate: br_j: the blink rate for any minute j
- Standard/normal blink rate: br_{std} the normal blink rate i.e 15 to 20 blinks/minute
- Type: app_{type}: the type of application (work=1/ entertainment=0)
- Posture (p): user's posture while using the application. Posture is not considered if user is using entertainment application. If posture is left, right, all set, too far then value is 1, else if it is too down or too close then it will be -1. The posture will be too down or close when user is stressed or tired. The posture can be relaxed when user is using entertainment applications, so it will be cancelled out as apptype value will be 0.
- If the blink-rate > std rate, then the value will be positive or else it will be negative.
- If posture is not correct while using work application , then posture will be -1.
- If blink rate and posture both are not incorrect then the value will be negative. The more value is negative indicates that the more user will be stressed.

$$stress = \sum_{i=0}^{total_{apps}} \sum_{j=0}^{a_i duration} (br_j - br_{std}/100) + (app_{type} \times p) \quad (7)$$

5. Experiments and Results

The proposed methodology was implemented and evaluated at the real time by taking the current data from students of *COMSATS*[1] University, Pakistan. The experimentation has been conducted on some students by executing the .exe file on different laptop machines. The configuration detail is tabulated below:

TABLE 1. HARDWARE REQUIREMENTS

RAM	Memory	Processor	Operating System	Load Time
8GB	128GB	Intel core i5	Windows 10	5 sec
4GB	400GB	Intel core m	Windows 8	10 sec

The graph generated by testing one of the students for monitoring the activity is given in Figure 7. This shows the red bars for work applications e.g. IDE and green bars for entertainment applications such as MS paint, media player etc. The graph in Figure 8 illustrates blink per minute. The graph shows blinks on y-axis and time on y-axis. The graph allows user to visually interpret his blinking patterns. The green line shows the threshold value that is 20 blinks per minute for an average human being. The graph in Figure 9 is demonstrating posture. The x-axis depicts the posture parameters being measured and y-axis shows their frequency. The graph is vital to understand posture trends of the user over the course of computer usage.

Based on the experiments conducted on students, a classification model was generated. The Na¨ıve Bayes Classification model was used to estimate the probability of stress in user. The classification model specifies whether a user can be tired or stressed based on input parameters.

TABLE 2. CLASSIFIER SAMPLE TABLE

Sr.	Blink rate	App type	Posture	Time used (in minutes)	Tired
1	10	work	too close	45	yes
2	15	work	too left	30	yes
3	21	work	all set	15	no
4	17	entertainment	too far	25	no
5	13	work	too close	20	yes
6	20	entertainment	too far	15	no
7	25	entertainment	too far	10	no
8	14	work	all set	35	yes
9	15	work	too down	15	yes
10	23	entertainment	too down	60	yes

The Table 2 is sample data generated based on Blink rate, app type, posture and time calculated and measured using our application; whereas, tired/stressed parameter was specified by the user. The applications that were monitored and their classification is given in Table 3.

In order to create the Na¨ıve Bayes Classifier, the counts relevant to each attribute value in a category are calculated. There are two types of data: categorical and continuous. Application type and posture fall in categorical type and blink rate and time used per application is continuous data. For categorical data count of yes/no were simply listed;

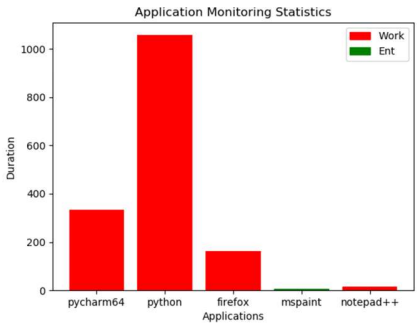

Figure 7. Application monitoring graph

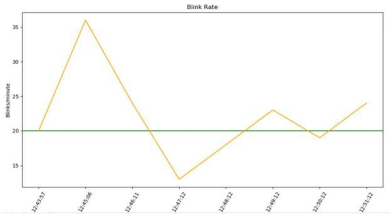

Figure 8. Blink rate graph

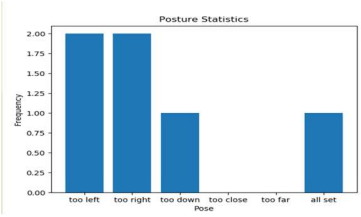

Figure 9. Posture statistics graph

TABLE 3. APPLICATION CLASSIFIER

App name	App type
MS Word	work
MS Excel	work
PyCharm	work
Facebook	entertainment
Youtube	entertainment
MS paint	entertainment
Eclipse	work

whereas, mean (μ), variance (σ^2), and standard deviation (σ) was calculated for continuous data. The formula to calculate the conditional probability of an attribute A with value 'a' under condition 'c' is given as in Equation 8:

$$P(A_i = a \| C = c) = N_{ac} / N_c \quad (8)$$

The formula to calculate overall conditional probability works by multiplying all conditional probabilities together. If any of them is 0, the whole expression becomes 0. There- fore, Laplace smoothing was applied to Posture attribute as it contains 0 values. The formula of Laplace Smoothing is given in Equation 9:

$$P(A_i = a \| C = c) = N_{ac} + 1 / N_c + N_i \quad (9)$$

Where N_i represents the number of attribute values for attribute A_i.

The Table 4 was generated after calculating the conditional probability and Table 5 shows data after applying Laplace smoothing.

TABLE 4. NAIVE BAYES CLASSIFIER

		yes=6	no=4
3* Blink rate	μ	15	15.7
	σ^2	10.4	8.2
	σ	3.224903	2.863564
2* App type	work	5	1
	ent	1	3
6* Posture	too close	2	0
	too far	0	3
	too left	1	0
	too right	0	0
	too down	2	0
	all set	1	1
3*Time used	μ	34.2	16.25
	σ^2	228.5	29.7
	σ	15.11622	5.449771

TABLE 5. CLASSIFIER WITH LAPLACE SMOOTHING

		yes=6	no=4
3*Blink rate	μ	15	15.7
	σ^2	10.4	8.2
	σ	3.224903	2.863564
2*App type	work	0.83333	00.25
	ent	0.166667	0.75
6*Posture	too close	50.2	0.1
	too far	0.083333	0.4
	too left	0.166667	0.1
	too right	0.08333	0.1
	too down	0.25	0.1
	all set	0.166667	0.2
3*Time used	μ	34.2	16.25
	σ^2	228.5	29.7
	σ	15.11622	5.449771

The conditional properties of each attribute were calculated with respect to class yes and no. The formula to calculate conditional probability for continuous attribute is given below:

$$P(A_i = a \| C = c) = 1/\sqrt{2\mu(\sigma)^2} e^{-(a-\mu)\,\zeta/2\sigma^{\zeta}} \quad (10)$$

The conditional properties for Class 'yes' and 'no' are given in Table 6:

TABLE 6. CONDITIONAL PROBABILITIES

Class	yes	no
Blink rate	0.2	0.06
App type	0.83	0.25
Posture	0.25	0.1
Time used	0.024	4.68×10^{-6}
Overall	9.96×10^{-4}	7.02×10^{-9}

As $P(X|Yes) > P(X|No)$ so the given record falls in the class Yes.

6. Evaluation

We evaluated the working of FitAssist with a commercially available product Logtale. For this purpose, FitAs- sist and Logtale were installed on the same system. Both applications cannot work simultaneously as both require webcam. Logtale does not require initial posture calibration. It monitors user for 1 to 2 minutes to evaluate user's posture. FitAssist requires user to manually calibrate posture by sitting in his preferred position and set posture. Logtale posture statistics are shown in Figure **10**. The application classifies posture as either good or bad. It does not provide any exercise recommendations based on poor posture. FitAssist uses posture statistics to determine whether the user needs exercise recommendations or not.

Logtale keeps track of work session and break based on presence of user in front of the computer screen. If a user is present in front of the screen, irrespective of the application he/she uses, it will be considered as a work session. Fig- ure 10 shows usage statistics generated by Logtale, showing work and break session over a period of days. FitAssist monitors the user presence and the type of application he/she is using to assess whether he/she is stressed or not. This approach helps to find out the time user was actually working on computer. The idle time or break session is however considered same in both the applications. If the user is not detected by the webcam, then it counts as idle time. The Table 7 below compares FitAssist with Logtale based on features such as posture, application, and usage monitoring. Types of notifications shown and technique used to evaluate usage is also listed. The comparison clearly shows that FitAssist provides more features. It can provide better statistical results related to posture and application usage.

7. Conclusion and Futurework

Despite the fact that many fitness gadgets and applications are already in use and doing a massive business, "FitAssist" will be a trend maker if published with all the goals and objectives as mentioned. The basic objective of this study is to develop a system that would monitor the posture of the users while they are working on computer.

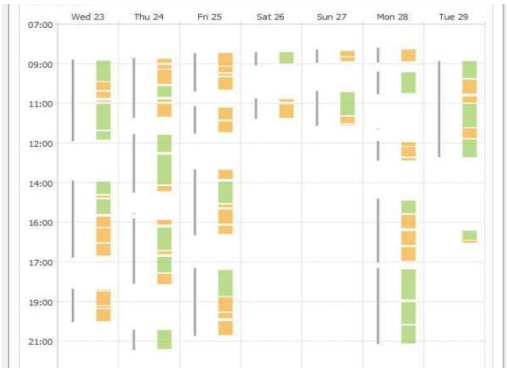

Figure 10. Logtale computer usage statistics

TABLE 7. COMPARISON TABLE

Feature	FitAssit	LogTale
Facial recognition	yes	no
Posture monitoring	yes	yes
Posture calibration re- quired	yes	no
Break reminders	yes	yes
Blink rate	yes	no
Stress monitoring	yes	no
Exercise recommen- dations	yes	no
Statistics duration	2 day	number of days spec- ified by user
Types of posture de- tected	close/far/left/right/dow n	good/bad
Posture notifications	yes	yes
Blink notifications	yes	no
Stress notifications	yes	no
Types of notifications	toast, sound based or both	toast notifications
Technique used for usage statistics	based on applications used by the user	based on pomodoro technique

The application regularly alerts its users about their posture, blink rate and usage limits. Wherever any of these measures are out of threshold values, the system alerts the users. This continuously reminds user to sit in correct posture and take regular breaks from work. Consequently, it helps computer users to know more about their activities while at work and helps improve them over time. “FitAssist” is users friendly due to its features and ease of operating. Its compact size makes it aesthetically attractive as well. It fulfills the everlasting demand of the people to stay fit while working on the computers for longer hours. The future work includes to introduce the “FitAssist” as a commercial product by improving accuracy and reducing error rate. The application can be modified to store user data for more than a day. This will help to detect trends of application usage and posture patterns over an extended period of time.

References

[1] Lin F, Lin W. The development of a multi-posture sensing system. In: 2014 7th International Conference on BioMedical Engineering and Informatics; Dalian, China; 2014. pp. 597-601.

[2] Zonouz R, Tehran H, Rahmani R. Smart Phone centric human posture monitoring system. In: 2014 IEEE Canada International Humanitarian Technology Conference - (IHTC); Montreal, QC, Canada; 2014. pp. 1-4.

[3] Lin W, Lee M, Chou WC. The design and development of a wearable posture monitoring vest. In: IEEE International Conference on Con- sumer Electronics (ICCE); Las Vegas, NV, USA; 2014. pp. 329-330.

[4] Matar G, Lina M, Carrier J, Riley A, Kaddoum G. Internet of things in sleep monitoring: An Application for posture recognition using supervised learning. In: IEEE 18th International Conference on e- Health Networking, Applications and Services (Healthcom); Munich, Germany; 2016. pp. 1-6.

[5] Shin J, Kang B, Park T, Huh J, Kim J et al. BeUpright: Posture Cor- rection Using Relational. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems; San Jose, CA, USA; 2016. pp. 6040-6052.

[6] Wu YP, Chen JH. A SURVEILLANCE SYSTEM DESIGNED FOR THE CORRECTION OF SITTING POSTURE. In: 9th International Conference on Ubiquitous Intelligence and Computing and 9th Inter- national Conference on Autonomic; Fukuoka, Japan; 2012. pp. 771- 773.

[7] Jiang H, Lee ZN, Drew MS. Recognizing Posture in Pictures with Successive Convexification and Linear Programming. IEEE Computer Society 2007; 14(2): 26-37. doi: 10.1109/MMUL.2007.41

[8] Moldoveanu A, Moldoveanu F,Butean A. Technologies For Moni- toring Students Body Posture and Physiological Parameters during Learning. In: The 10th International Scientific Conference eLearning and software for Education, Bucharest; Bucharest, Romania; 2014. pp. 58-62.

[9] Soukupov T, Cech J. Real-Time Eye Blink Detection using Facial Landmarks. In: 21st Computer Vision Winter Workshop; Rimske Toplice, Slovenia; 2016.