# CONTENTS PAGE.NO

1. INTRODUCTION 1
2. LITERATURE SURVEY 4
   1. [EXISTING SYSTEM](#_TOC_250006)
   2. [PROPOSED SYSTEM](#_TOC_250005)
3. SYSTEM MODELS AND NOTATIONS 8
   1. ARCHITECTURE
   2. [SYSTEM DESIGN](#_TOC_250004)
   3. MODULES
4. SOFTWARE DESIGNS 14
   1. [CLASS DIAGRAM](#_TOC_250003)
   2. [SEQUENCE DIAGRAM](#_TOC_250002)
   3. [ACTIVITY DIAGRAM](#_TOC_250001)
5. SYSTEM STUDY 17
6. SYSTEM TESTING 19
7. SOFTWARE AND HARDWARE REQUIREMENTS 22
8. SCREENSHORTS 23
9. SOFTWARE ENVIRONMENT 40
   1. PYTHON
   2. [DJANGO](#_TOC_250000)
10. CONCLUSION 69
11. REFERENCES 70

INTRODUCTION:

# CHAPTER 1

Stress management systems play a significant role to detect the stress levels which disrupts our socio economic lifestyle. As World Health Organization (WHO) says, Stress is a mental health problem affecting the life of one in four citizens. Human stress leads to mental as well as socio-fiscal problems, lack of clarity in work, poor working relationship, depression and finally commitment of suicide in severe cases. This demands counselling to be provided for the stressed individuals cope up against stress. Stress avoidance is impossible but preventive actions helps to overcome the stress. Currently, only medical and physiological experts can determine whether one is under depressed state (stressed) or not. One of the traditional method to detect stress is based on questionnaire.

This method completely depends on the answers given by the individuals, people will be tremulous to say whether they are stressed or normal. Automatic detection of stress minimizes the risk of health issues and improves the welfare of the society. This paves the way for the necessity of a scientific tool, which uses physiological signals thereby automating the detection of stress levels in individuals. Stress detection is discussed in various literatures as it is a significant societal contribution that enhances the lifestyle of individuals. Ghaderi et al. analysed stress using Respiration, Heart rate (HR), facial electromyography (EMG), Galvanic skin response (GSR) foot and GSR hand data with a conclusion that, features pertaining to respiration process are substantial in stress detection. Maria Viqueira et al. describes mental stress prediction using a standalone stress sensing hardware by interfacing GSR as the only physiological sensor David Liu et al. proposed a research to predict stress levels solely from Electrocardiogram (ECG). Multimodal sensor efficacy to detect stress of working people is experimentally discussed in . This employs the sensor data from sensors such as pressure distribution, HR,Blood Volume Pulse (BVP) and Electrodermal activity (EDA).

An eye tracker sensor is also used which systematically analyses the eye movements with the stressors like Stroop word test and information related to pickuptasks.The authors of performed perceived stress detection by a set of non-invasive sensors which collects the physiological signals such as ECG , GSR, Electroencephalography (EEG), EMG, and Saturation of peripheral oxygen (SpO2). Continuous stress levels are estimated using the physiological sensor data such as GSR, EMG, HR, Respiration in. The stress detection is carried out effectively using Skin conductance level (SCL), HR, Facial EMG

sensors by creating ICT related Stressors. Automated stress detection is made possible by several pattern recognition algorithms. Every sensor data is compared with a stress index which is a threshold value used for detecting the stress level.

The authors of collected data from 16 individuals under four stressor conditions which were tested with Bayesian Network, J48 algorithm and Sequential Minimal Optimization (SMO) algorithm for predicting stress. Statistical features of heart rate, GSR , frequency domain features of heart rate and its variability (HRV), and the power spectral components of ECG were used to govern the stress levels. Various features are extracted from the commonly used physiological signals such as ECG, EMG, GSR, BVP etc., measured using appropriate sensors and selected features are grouped into clusters for further detection of anxiety levels . In, it is concluded that smaller clusters result in better balance in stress detection using the selected General Regression Neural Network (GRNN) model. This results in the fact that different combinations of the extracted features from the sensor signals provide better solutions to predict the continuous anxiety level. Frequency domain features like LF power (low frequency power from 0.04 Hz to0.15Hz), HF power (High frequency power from 0.15Hz to 0.4 Hz) , LF/HF (ratio of LF to the HF ). and time domain features like Mean , Median, standard deviation of heart signal are considered for continuous real time stress detection in . Classification using decision tree such as PLDA is performed using two stressors namely pickup task and stroop based word test wherein the authors concluded that the stressor based classification proves unsatisfactory. In 2016, Gjoreski et al. created laboratory based stress detection classifiers from ECG signal and HRV features. Features of ECG are analysed using GRNN model to measure the stress level. Heart rate variability (HRV) features and RR (cycle length variability interval length between two successive Rs) interval features are used to classify the stress level. It is noticed that Support Vector Machine (SVM) was used as the classification algorithm predominantly due to its generalization ability and sound mathematical background Various kernels were used to develop models using SVM and it is concluded in that a linear SVM on both ECG frequency features and HRV features performed best, outperforming other model choices.

Nowadays as IT industries are setting a new peek in the market by bringing new technologies and products in the market. In this study, the stress levels in employees are also noticed to raise the bar high. Though there are many organizations who provide mental health related schemes for their employees but the issue is far from control. In this paper we try to go in the depth of this problem by trying to detect the stress patterns in the working employee in the companies we would like to apply image processing and

machine learning techniques to analyze stress patterns and to narrow down the factors that strongly determine the stress levels. Machine Learning algorithms like KNN classifiers are applied to classify stress.

Image Processing is used at the initial stage for detection, the employee’s image is clicked by the camera which serves as input. In order to get an enhanced image or to extract some useful information from it image processing is used by converting image into digital form and performing some operations on it. By taking input as an image from video frames and output may be image or characteristics associated with that image. Image processing basically includes the following three steps:

 Importing the image via image acquisition tools.  Analyzing and manipulating the image.

 Output in which result is altered image or report that is based on image analysis.

System gets the ability to automatically learn and improve from self-experiences without being explicitly programmed using Machine learning which is an application of artificial intelligence (AI). Computer programs are developed by Machine Learning that can access data and use it to learn for themselves. Explicit programming to perform the task based on predictions or decisions builds a mathematical model based on "training data" by using Machine Learning. The extraction of hidden data, association of image data and additional pattern which are unclearly visible in image is done using Image Mining. It’s an interrelated field that involves, Image Processing, Data Mining, Machine Learning and Datasets. According to conservative estimates in medical books, 50- 80% of all physical diseases are caused by stress. Stress is believed to be the principal cause in cardiovascular diseases. Stress can place one at higher risk for diabetes, ulcers, asthma, migraine headaches, skin disorders, epilepsy, and sexual dysfunction. Each of these diseases, and host of others, is psychosomatic (i.e., either caused or exaggerated by mental conditions such as stress) in nature. Stress has three prong effects:

 Subjective effects of stress include feelings of guilt, shame, anxiety, aggression or frustration. Individuals also feel tired, tense, nervous, irritable, moody, or lonely.

 Visible changes in a person's behavior are represented by Behavioral effects of stress. Effects of behavioral stress are seen such as increased accidents, use of drugs or alcohol, laughter out of context, outlandish or argumentative behavior, very excitable moods, and/or eating or drinking to excess.

 Diminishing mental ability, impaired judgment, rash decisions, forgetfulness and/or hypersensitivity to

criticism are some of the effects of Cognitive stress

## CHAPTER 2

**LITERATURE SURVEY :**

1. **Stress and anxiety detection using facial cues from videos AUTHORS: G. Giannakakis, D. Manousos, F. Chiarugi**

This study develops a framework for the detection and analysis of stress/anxiety emotional states through video-recorded facial cues. A thorough experimental protocol was established to induce systematic variability in affective states (neutral, relaxed and stressed/anxious) through a variety of external and internal stressors. The analysis was focused mainly on nonvoluntary and semi-voluntary facial cues in order to estimate the emotion representation more objectively. Features under investigation included eye-related events, mouth activity, head motion parameters and heart rate estimated through camerabased photoplethysmography. A feature selection procedure was employed to select the most robust features followed by classification schemes discriminating between stress/anxiety and neutral states with reference to a relaxed state in each experimental phase. In addition, a ranking transformation was proposed utilizing self reports in order to investigate the correlation of facial parameters with a participant perceived amount of stress/anxiety. The results indicated that, specific facial cues, derived from eye activity, mouth activity, head movements and camera based heart activity achieve good accuracy and are suitable as discriminative indicators of stress and anxiety.

1. **Detection of Stress Using Image Processing and Machine Learning Techniques AUTHORS: Nisha Raichur, Nidhi Lonakadi, Priyanka Mural**

Stress is a part of life it is an unpleasant state of emotional arousal that people experience in situations like working for long hours in front of computer. Computers have become a way of life, much life is spent on the computers and hence we are therefore more affected by the ups and downs that they cause us. One cannot just completely avoid their work on computers but one can at least control his/her usage when being alarmed about

him being stressed at certain point of time. Monitoring the emotional status of a person

who is working in front of a computer for longer duration is crucial for the safety of a person. In this work a realtime non-intrusive videos are captured, which detects the emotional status of a person by analysing the facial expression. We detect an individual emotion in each video frame and the decision on the stress level is made in sequential hours of the video captured. We employ a technique that allows us to train a model and analyze differences in predicting the features. Theano is a python framework which aims at improving both the execution time and development time of the linear regression model which is used here as a deep learning algorithm. The experimental results show that the developed system is well on data with the generic model of all ages.

1. **Machine Learning Techniques for Stress Prediction in Working Employees AUTHORS: U. S. Reddy, A. V. Thota and A. Dharun**

Stress disorders are a common issue among working IT professionals in the industry today. With changing lifestyle and work cultures, there is an increase in the risk of stress among the employees. Though many industries and corporates provide mental health related schemes and try to ease the workplace atmosphere, the issue is far from control. In this paper, we would like to apply machine learning techniques to analyze stress patterns in working adults and to narrow down the factors that strongly determine the stress levels. Towards this, data from the OSMI mental health survey 2017 responses of working professionals within the tech-industry was considered. Various Machine Learning techniques were applied to train our model after due data cleaning and preprocessing. The accuracy of the above models was obtained and studied comparatively. Boosting had the highest accuracy among the models implemented. By using Decision Trees, prominent features that influence stress were identified as gender, family history and availability of health benefits in the workplace. With these results, industries can now narrow down their approach to reduce stress and create a much comfortable workplace for their employees.

1. **Classification of acute stress using linear and non-linear heart rate variability analysis derived from sternal ECG**

**AUTHORS** : **Tanev, G., Saadi, D.B., Hoppe, K., Sorensen, H.B**

Chronic stress detection is an important factor in predicting and reducing the risk of cardiovascular disease. This work is a pilot study with a focus on developing a method for detecting short-term psychophysiological changes through heart rate variability (HRV) features. The purpose of this pilot study is to establish and to gain insight on a set of features that could be used to detect psychophysiological changes that occur during chronic stress. This study elicited four different types of arousal by images, sounds, mental tasks and rest, and classified them using linear and non-linear HRV features from electrocardiograms (ECG) acquired by the wireless wearable ePatch® recorder. The highest recognition rates were acquired for the neutral stage (90%), the acute stress stage (80%) and the baseline stage (80%) by sample entropy, detrended fluctuation analysis and normalized high frequency features. Standardizing non-linear HRV features for each subject was found to be an important factor for the improvement of the classification results.

1. **HealthyOffice: Mood recognition at work using smartphones and wearable sensors**

### AUTHORS: Zenonos, A., Khan, A., Kalogridis, G., Vatsikas, S., Lewis, T., Sooriyabandara

Stress, anxiety and depression in the workplace are detrimental to human health and productivity with significant financial implications. Recent research in this area has focused on the use of sensor technologies, including smartphones and wearables embedded with physiological and movement sensors. In this work, we explore the possibility of using such devices for mood recognition, focusing on work environments. We propose a novel mood recognition framework that is able to identify five intensity levels for eight different types of moods every two hours. We further present a smartphone app ('HealthyOffice'), designed to facilitate self-reporting in a structured manner and provide our model with the ground truth. We evaluate our system in a small-scale user study where wearable sensing data is collected in an office environment. Our experiments exhibit promising results allowing us to reliably recognize various classes of perceived moods.

## EXISTING SYSTEM:

In the existing system work on stress detection is based on the digital signal processing, taking into consideration Galvanic skin response, blood volume, pupil dilation and skin temperature. And the other work on this issue is based on several physiological signals and visual features (eye closure, head movement) to monitor the stress in a person while he is working. However these measurements are intrusive and are less comfortable in real application. Every sensor data is compared with a stress index which is a threshold value used for detecting the stress level.

### DISADVANTAGES OF EXISTING SYSTEM:

* + - Physiological signals used for analysis are often pigeonholed by a Non-stationary time performance.
    - The extracted features explicitly gives the stress index of the physiological signals.The ECG signal is directly assessed by using commonly used peak j48 algorithm
    - Different people may behave or express differently under stress and it is hard to find a universal pattern to define the stress emotion.

**Algorithm**: Bayesian Network, J48

## PROPOSED SYSTEM:

The proposed System Machine Learning algorithms like KNN classifiers are applied to classify stress. Image Processing is used at the initial stage for detection, the employee’s image is given by the browser which serves as input. In order to get an enhanced image or to extract some useful information from it image processing is used by converting image into digital form and performing some operations on it. By taking input as an image and output may be image or characteristics associated with that images. The emotion are displayed on the rounder box. The stress level indicating by Angry, Disgusted, Fearful, Sad.

## ADVANTAGES OF PROPOSED SYSTEM:

* + - Output in which result is altered image or report that is based on image analysis.
    - Stress Detection System enables employees with coping up with their issues leading to stress by preventative stress management solutions.
    - We will capture images of the employee based on the regular intervals and then the tradition survey forms will be given to the employees

**Algorithm**: K-Nearest Neighbor (KNN)

# CHAPTER 3 SYSTEM MODELS AND NOTATIONS :

* 1. **SYSTEM ARCHITECTURE**

FACE ACQUITION AND PREPROCESSING MODULE



TRAINING

DATASET

MODEL

STRESS

DETECTION

STRESS DETECTION MODULE

OFFLINE DISPLACEMT CALCULATIO N

OFFLINE CLASSIFIER

VARIENCE OF DISPLACE MENT

STRESS DETEC

TION

EYEBROW COORDIN ATION

IMAGEPR E PROCESS ING

IMAGE ACQUITION

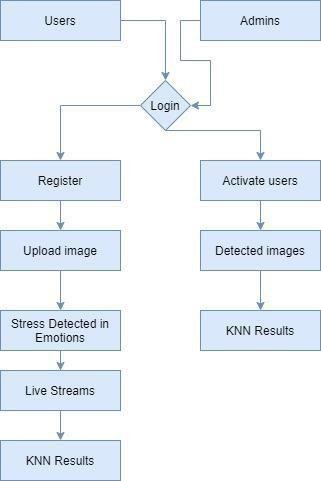
LEARNING THE LINEAR REGRESSION

ALGORITHM

# SYSTEM DESIGN

## DATA FLOW DIAGRAM:

* + 1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
    2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
    3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
    4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



## UML DIAGRAMS :

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

## GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

## USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Admin**

**Login**

**Upload Image**

**Stress Emotions**

**Live Stream**

**DeepLearning Live Stream**

**KNN Results**

**Activate users**

**Users**

## MODULES:

* **User**
* **Admin**
* **Data Preprocess**
* **Machine Learning MODULES DESCRIPTION:**

**User:**

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the customer. Once admin activated the customer then user can login into our system. First user has to give the input as image to the system. The python library will extract the features and appropriate emotion of the image. If given image contain more than one faces also possible to detect. The stress level we are going to indicate by facial expression like sad, angry etc..

The image processing completed the we are going to start the live stream. In the live stream also we can get the facial expression more that one persons also. Compare to tensorlflow live stream the tesnorflow live stream will fast and better results. Once done the we are loading the dataset to perform the knn classification accuracy precession scores.

.**Admin:**

Admin can login with his credentials. Once he login he can activate the users. The activated user only login in our applications. The admin can set the training and testing data for the project dynamically to the code. The admin can view all users detected results in hid frame. By clicking an hyperlink in the screen he can detect the emotions of the images. The admin can also view the knn classification detected results. The dataset in the excel format. By authorized persons we can increase the dataset size according the imaginary values.

**Data Preprocess:**

Dataset contains grid view of already stored dataset consisting numerous properties, by Property Extraction newly designed dataset appears which contains only numerical input variables as a result of Principal Component Analysis feature selection transforming to 6 principal components which are Condition (No stress, Time pressure, Interruption), Stress, Physical Demand, Performance and Frustration. **Machine Learning:**

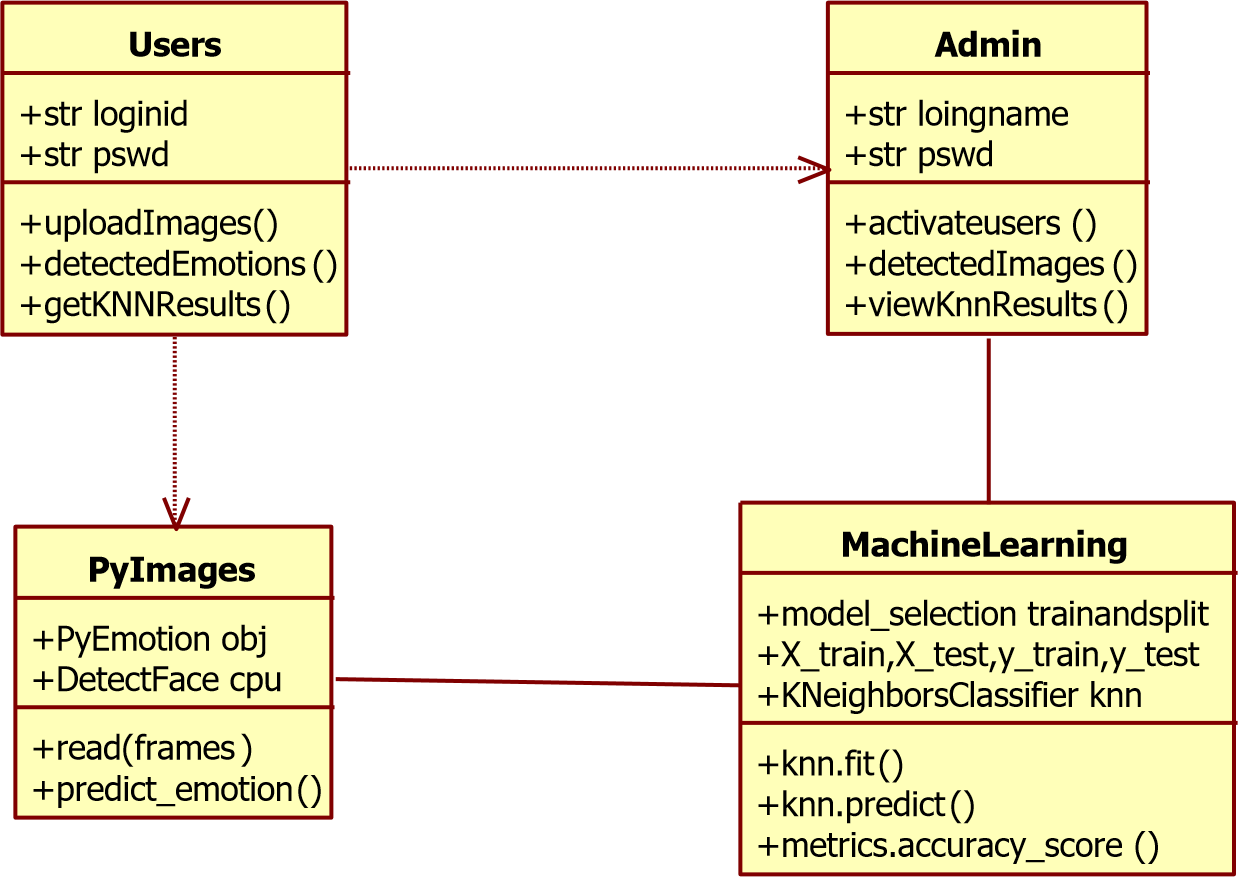
K-Nearest Neighbor (KNN) is used for classification as well as regression analysis. It is a supervised learning algorithm which is used for predicting if a person needs treatment or not. KNN classifies the dependent variable based on how similar it is; independent variables are to a similar instance from the already known data. the Knn Classification can be called as a statistical model that uses a binary dependent variable. In classification analysis, KNN is estimating the parameters of a KNN model. Mathematically, a binary KNN model has a dependent variable with two possible value, which is represented by an indicator variable, where the two values are labeled "0" and "1".

# CHAPTER 4

## SOFTWARE DESIGN:

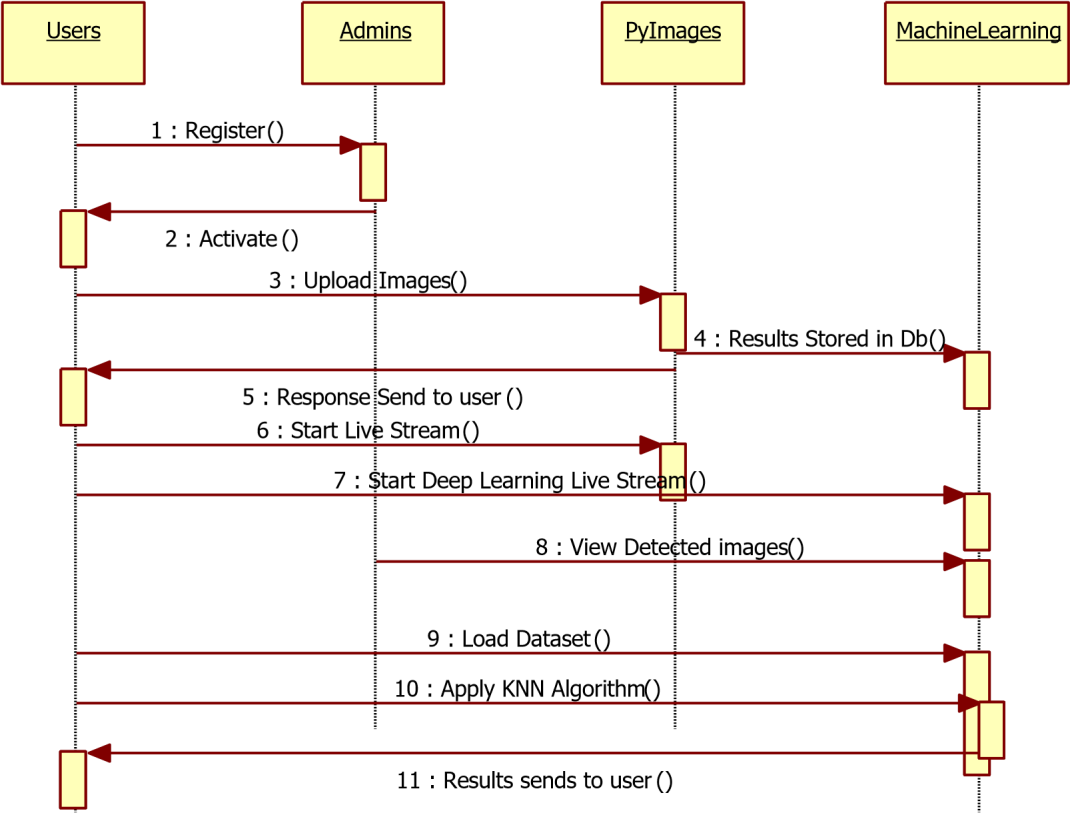
## CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



## SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



## ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-bystep workflows of components in a system. An activity diagram shows the overall flow of control.



Users

Admin

Upload Image

Activate users

Image Results

Detected images

Live Stream

Deep Learning Live Stream

KNN Results

KNN Results

# CHAPTER 5

## SYSTEM STUDY FEASIBILITY STUDY :

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

* + - ECONOMICAL FEASIBILITY
    - TECHNICAL FEASIBILITY
    - SOCIAL FEASIBILITY

## ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## SOCIAL FEASIBILITY :

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM TEST**

# CHAPTER 6

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## TYPES OF TESTS:

**Unit testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test:**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions,

or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test:**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing:**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing:**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach:**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives :**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* + Verify that the entries are of the correct format
  + No duplicate entries should be allowed
  + All links should take the user to the correct page.

### Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER 7 SOFTWARE AND HARDWARE REQUIREMENTS : HARDWARE REQUIREMENTS:**

* **System :** Intel i3.
* **Hard Disk :** 1 TB.
* **Monitor :** 14’ Colour Monitor.
* **Mouse :** Optical Mouse.
* **Ram :** 4GB.

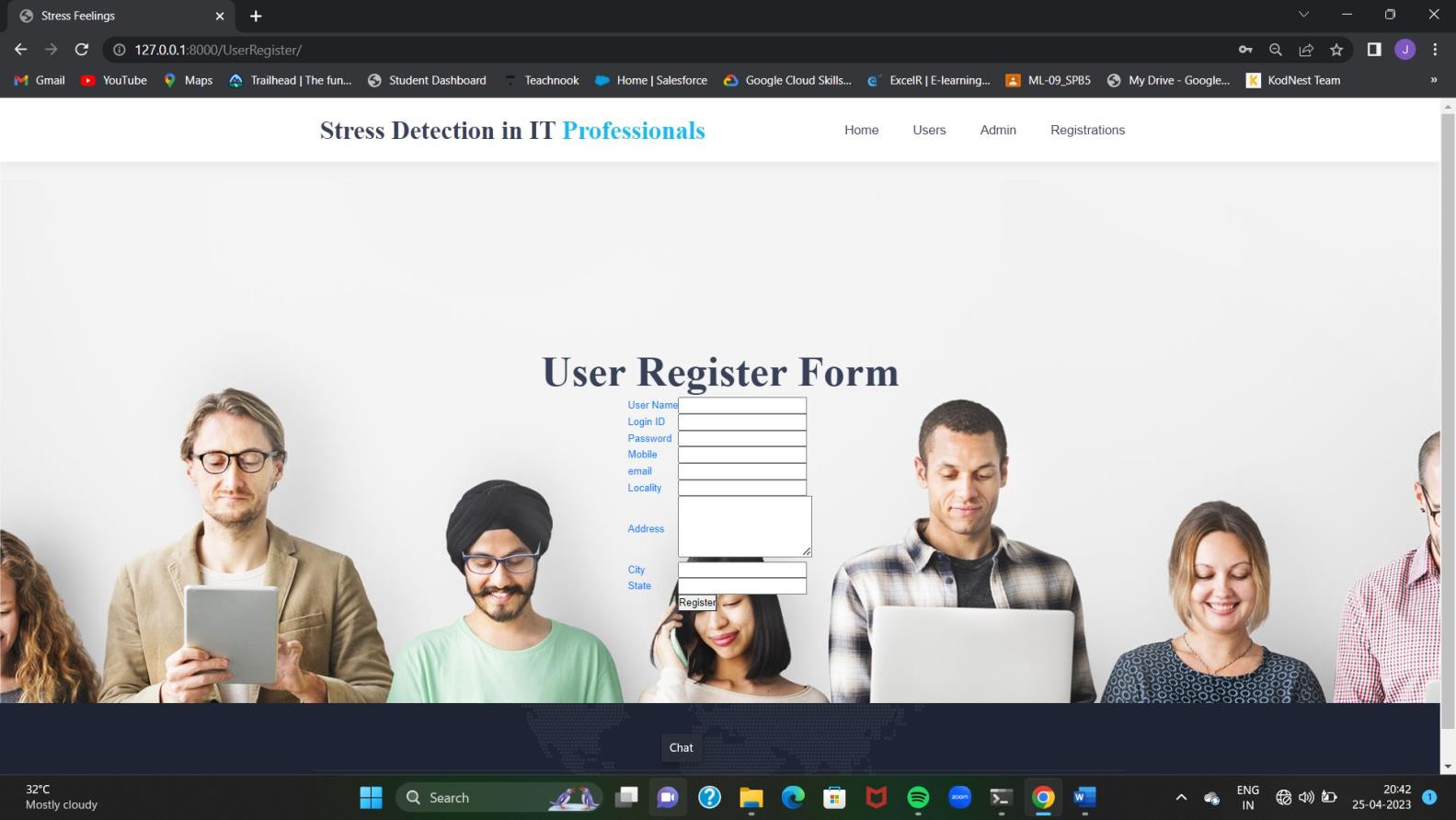
**SOFTWARE REQUIREMENTS:**

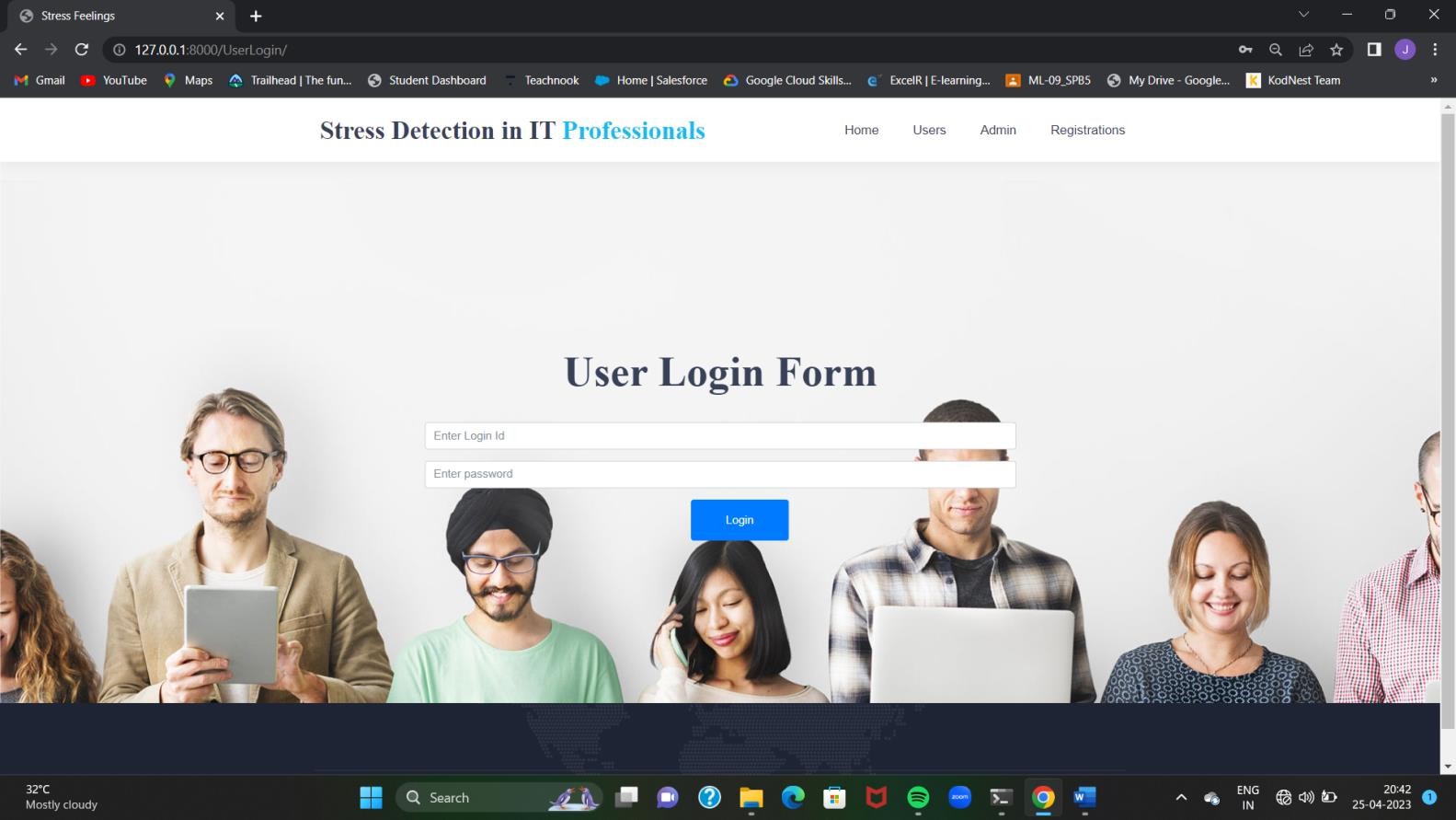
|  |  |
| --- | --- |
| * **Operating system** | **:** Windows 10. |
| * **Coding Language** | **:** Python. |
| * **Front-End** | **:** Html, CSS. |
| * **Designing** | **:** Html,css,javascript. |
| * **Data Base** | **:** SQLite. |

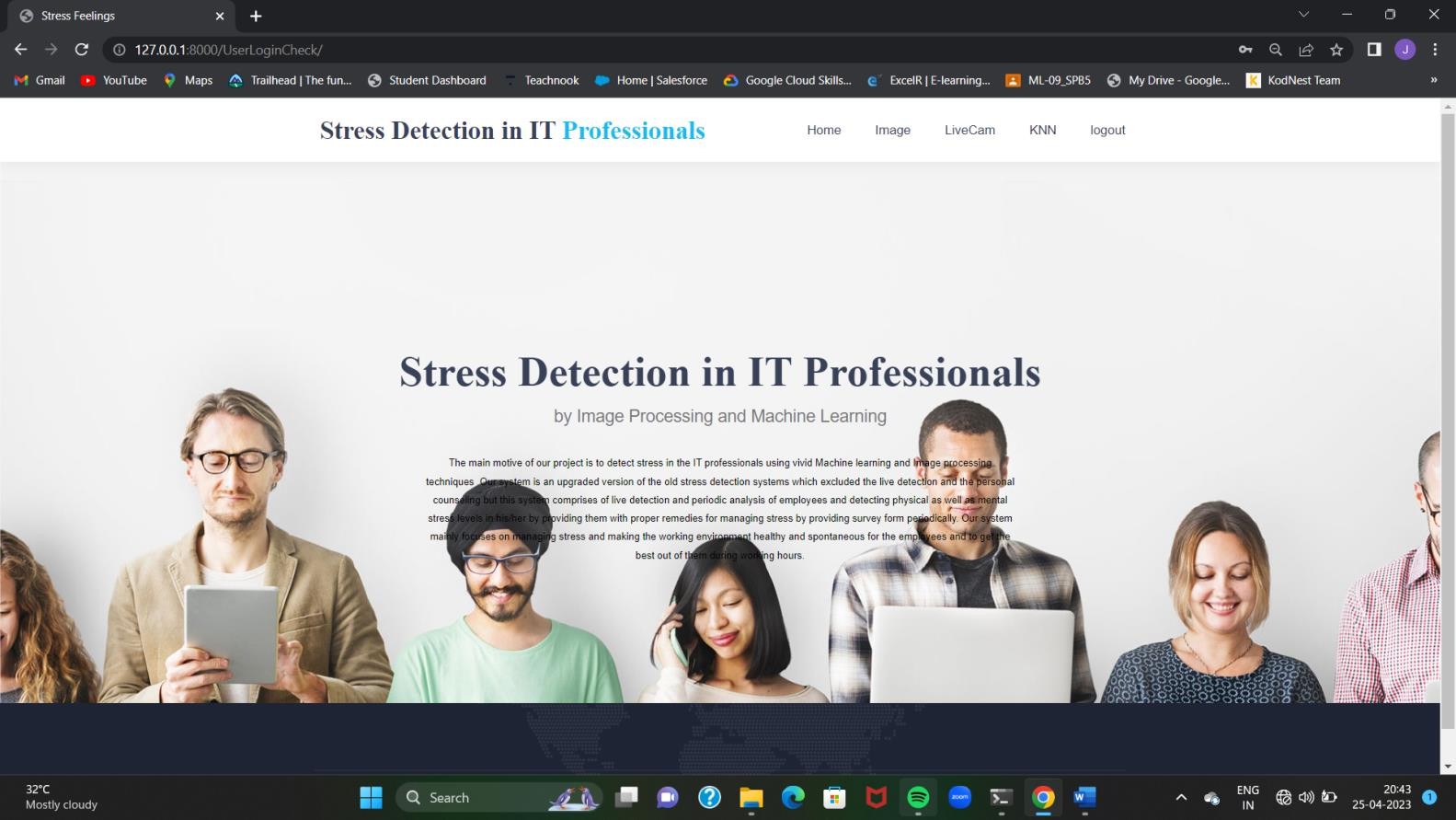
# CHAPTER 8

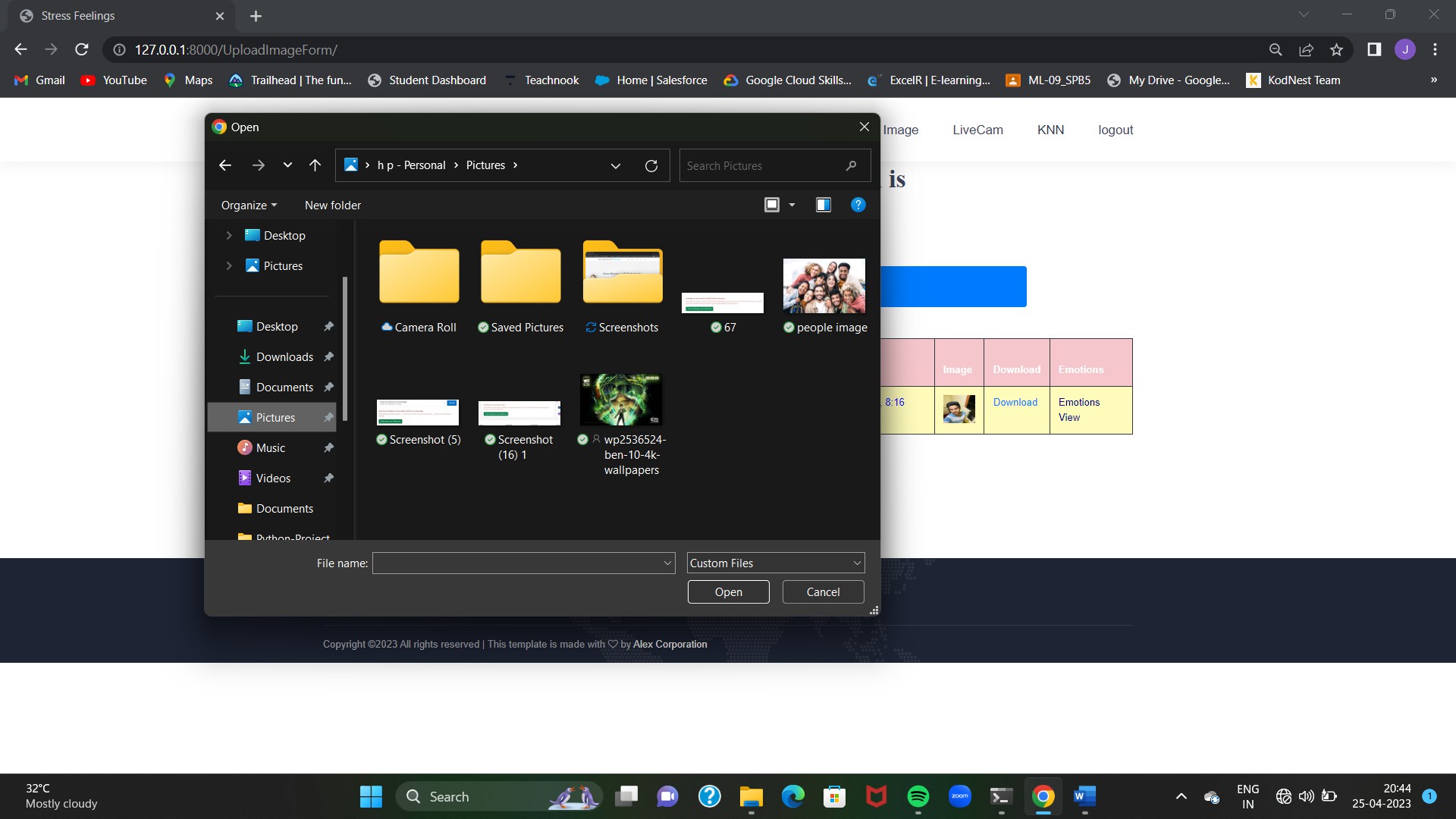
**SCREENSHOTS**

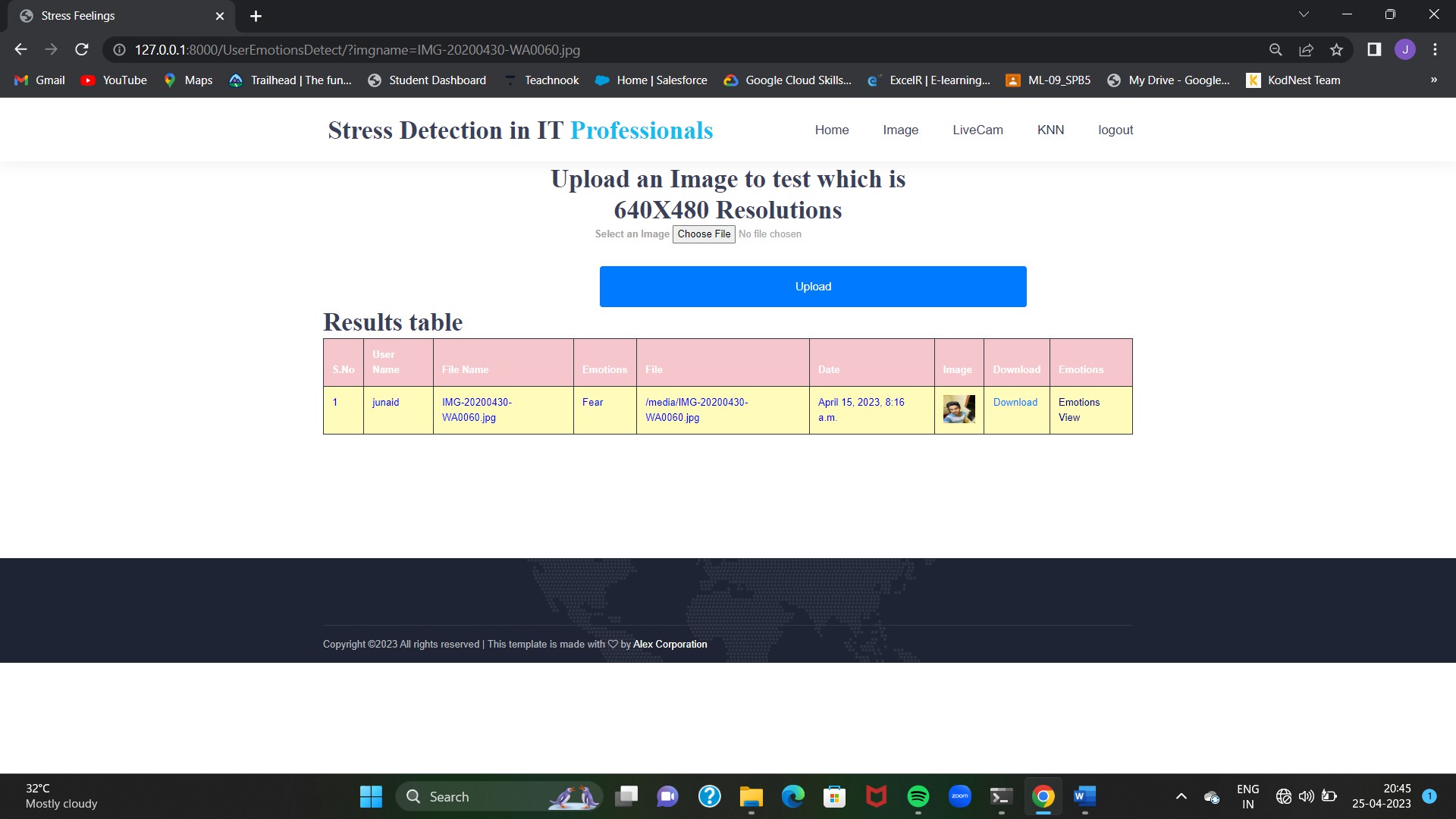
**Home page:**

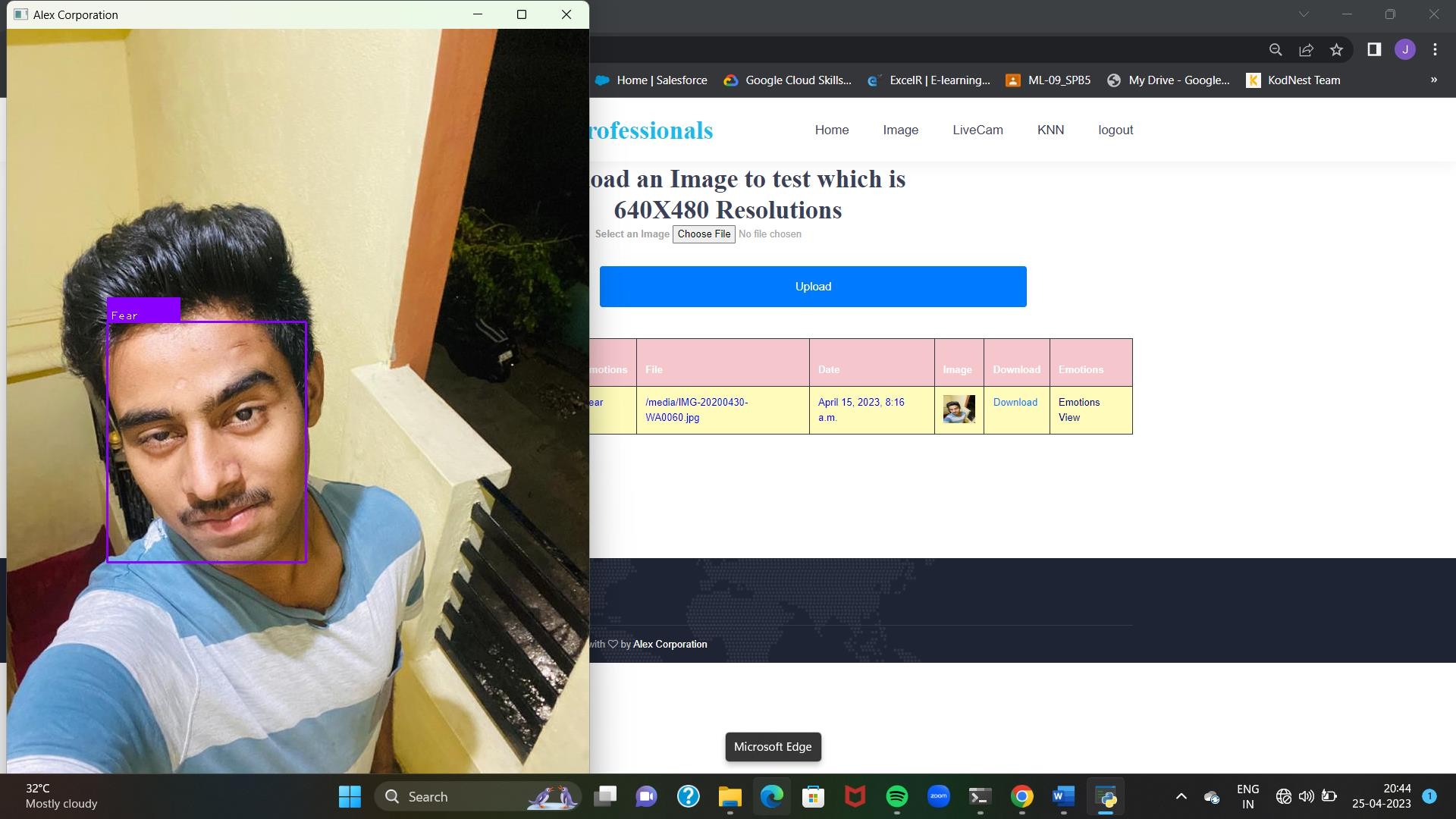
**User Register page:**

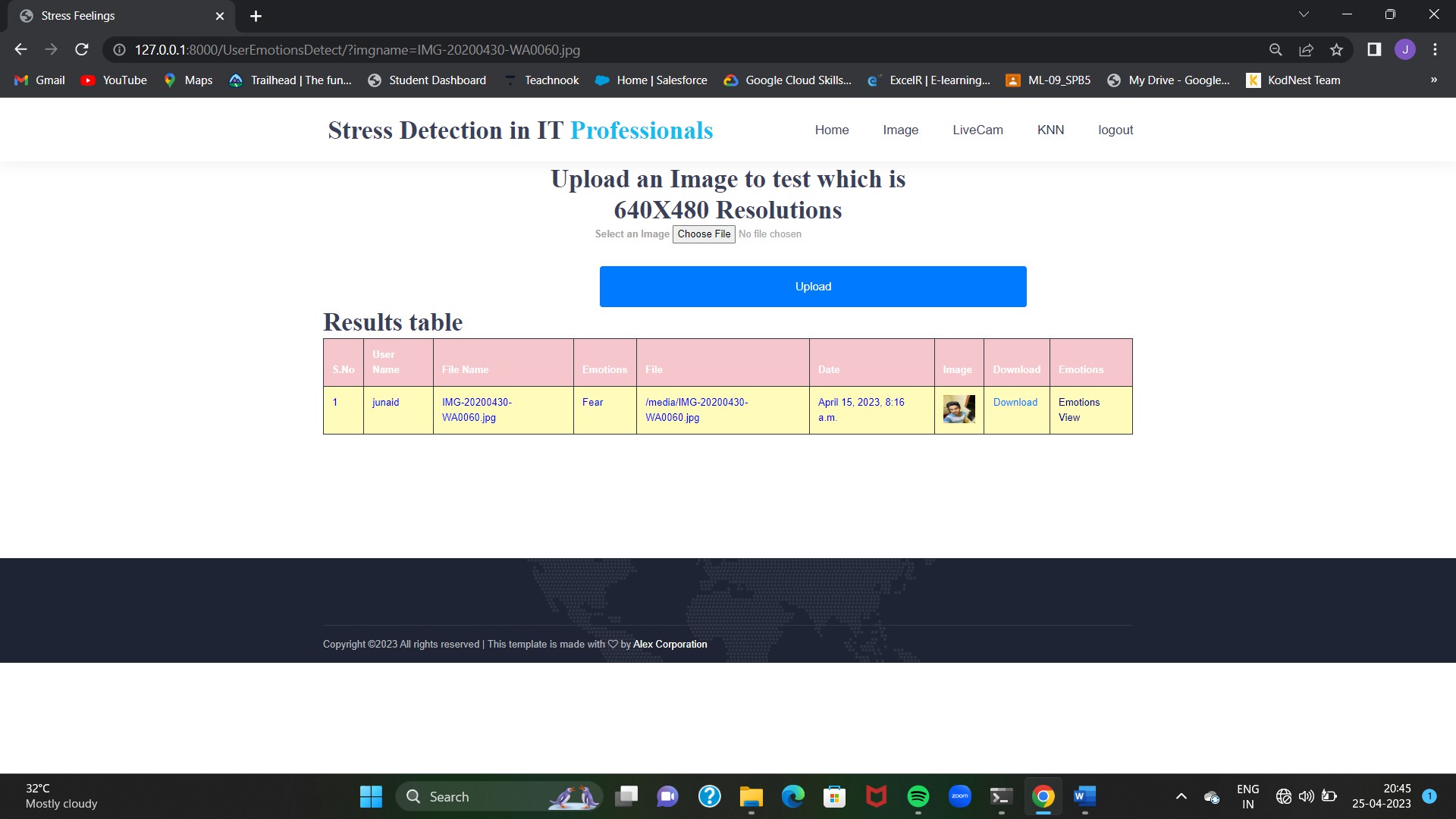
**User Login Form:**

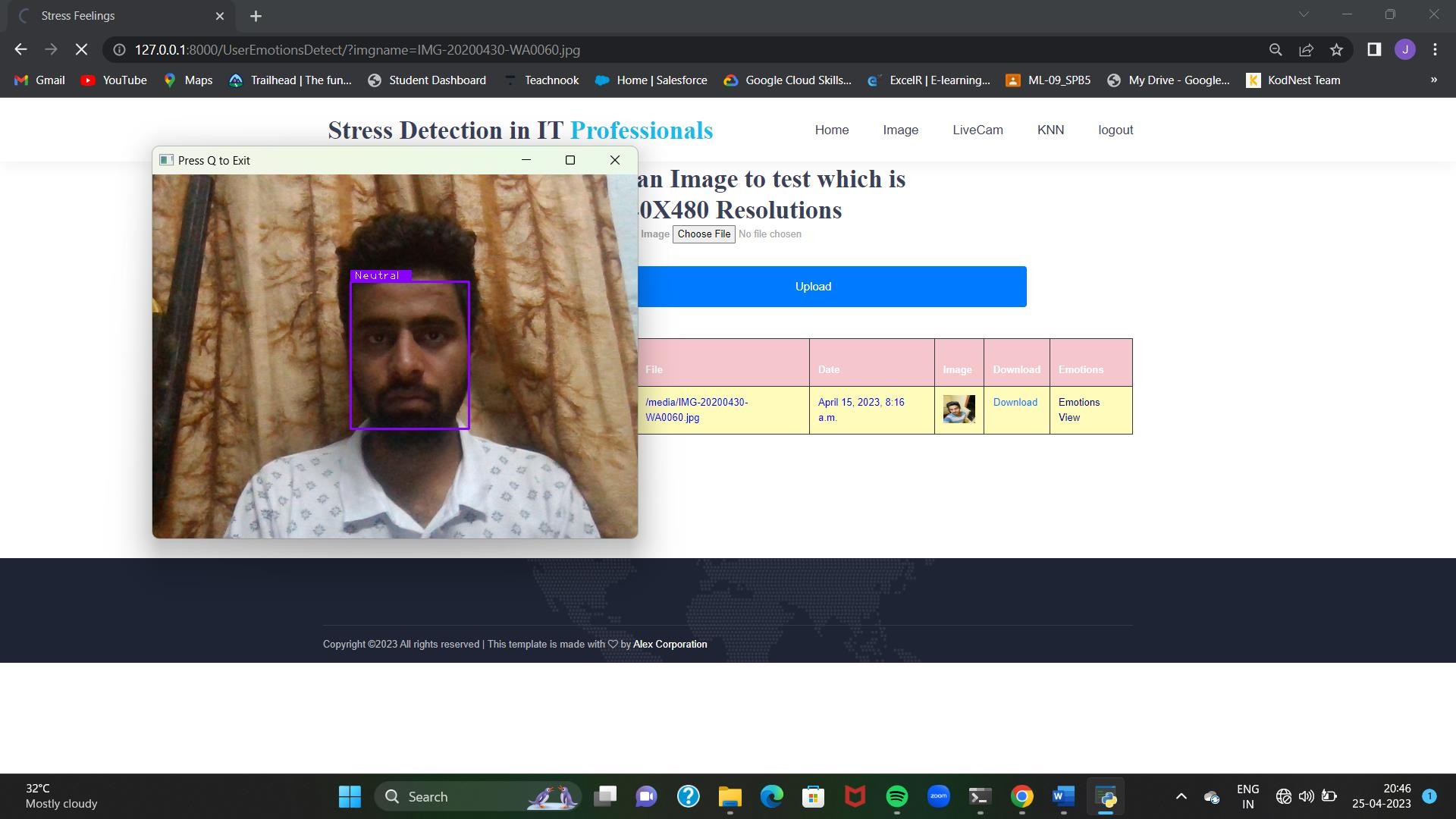
**User Home Page:**

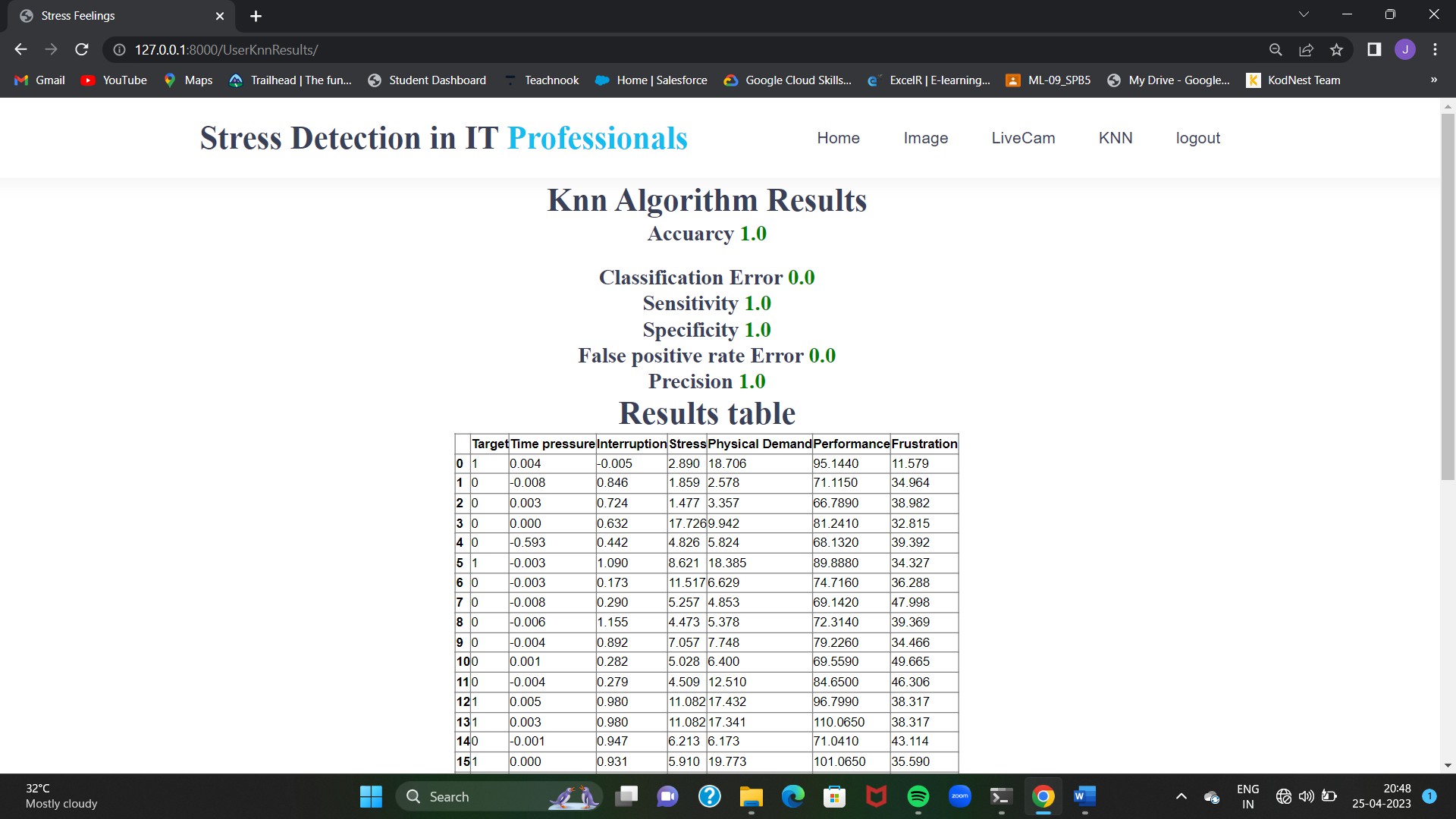
**Giving Image as Input:**

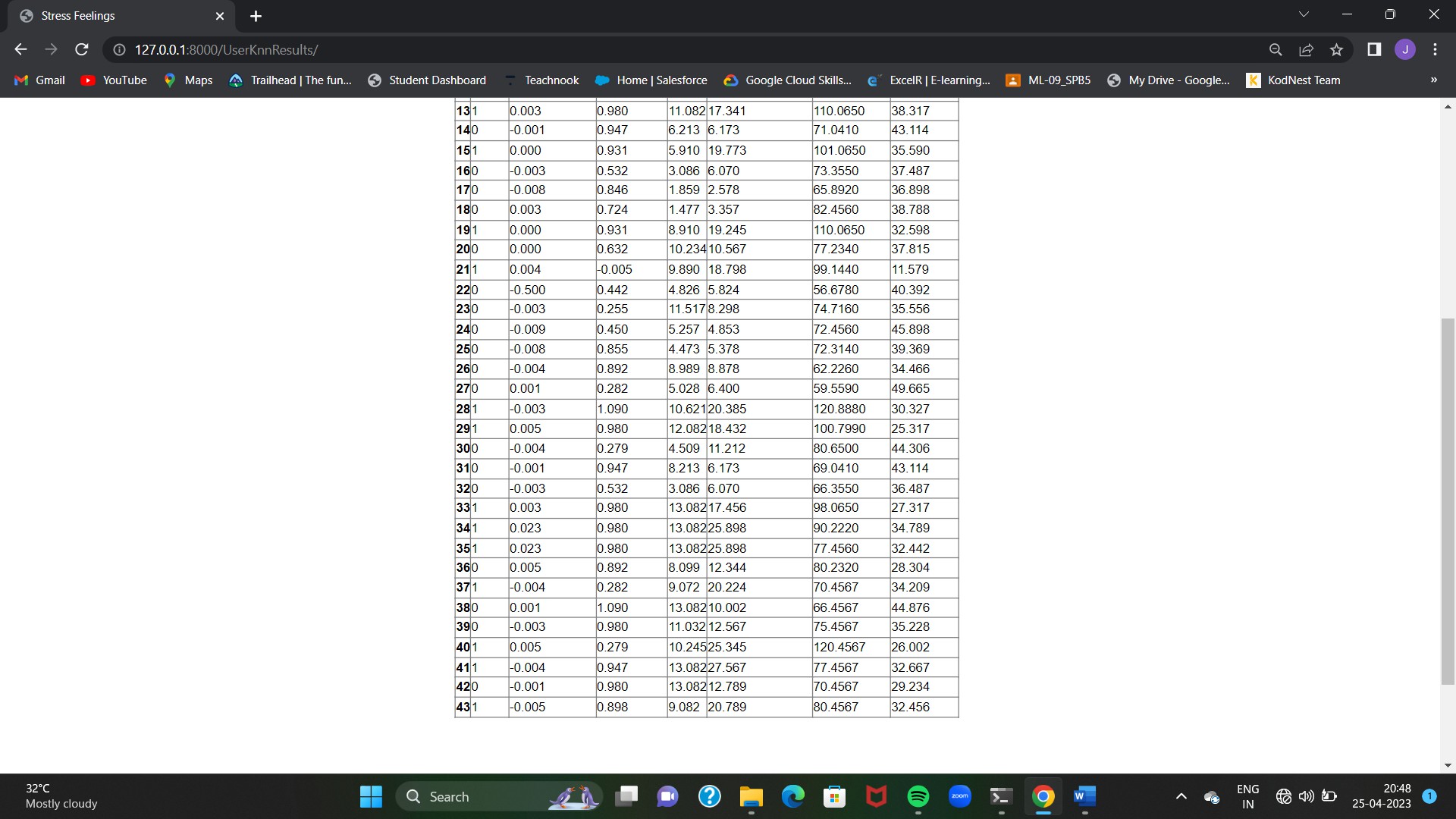
**Upload Image:**

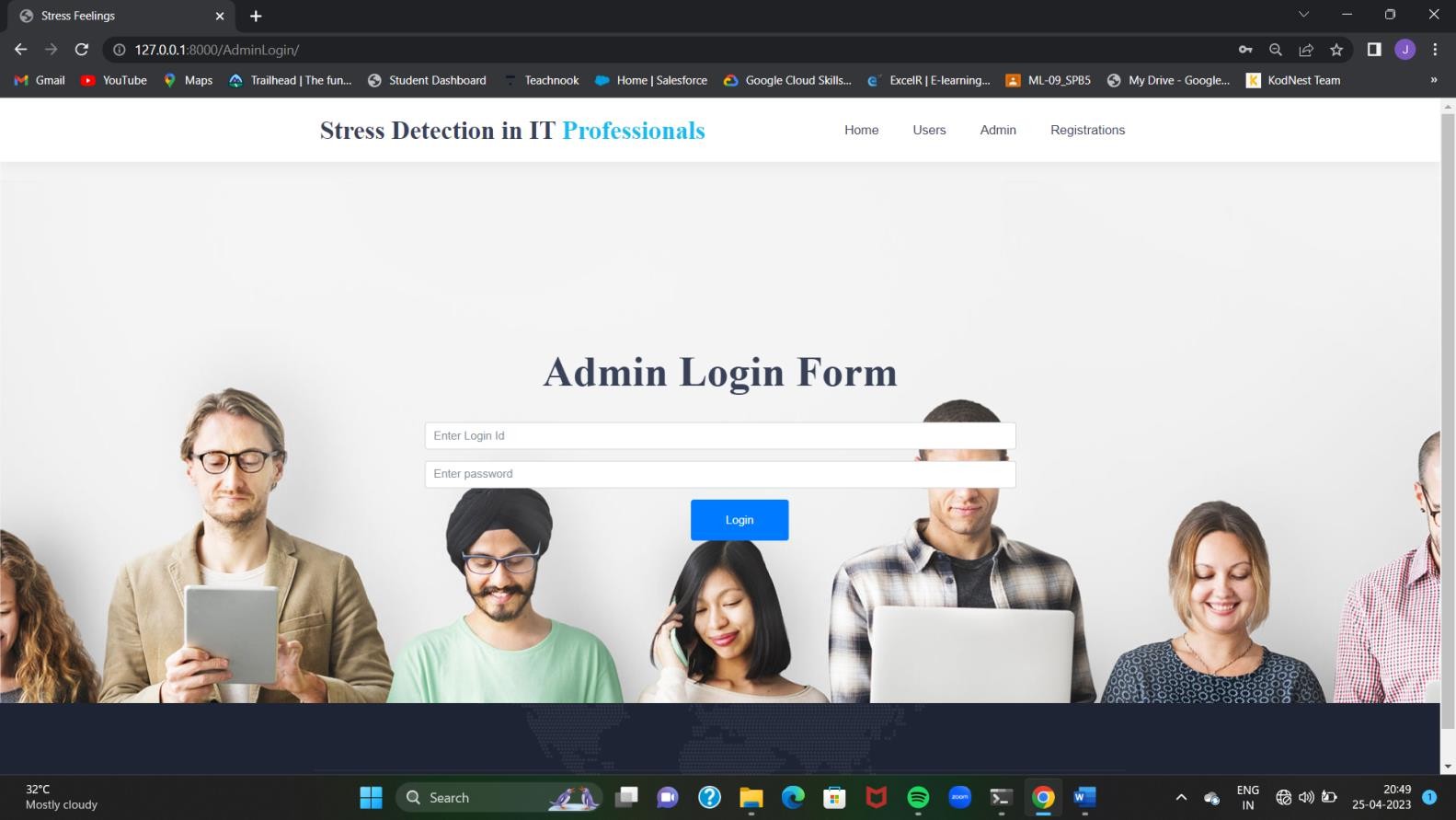
**Response Image:**

**Results:**

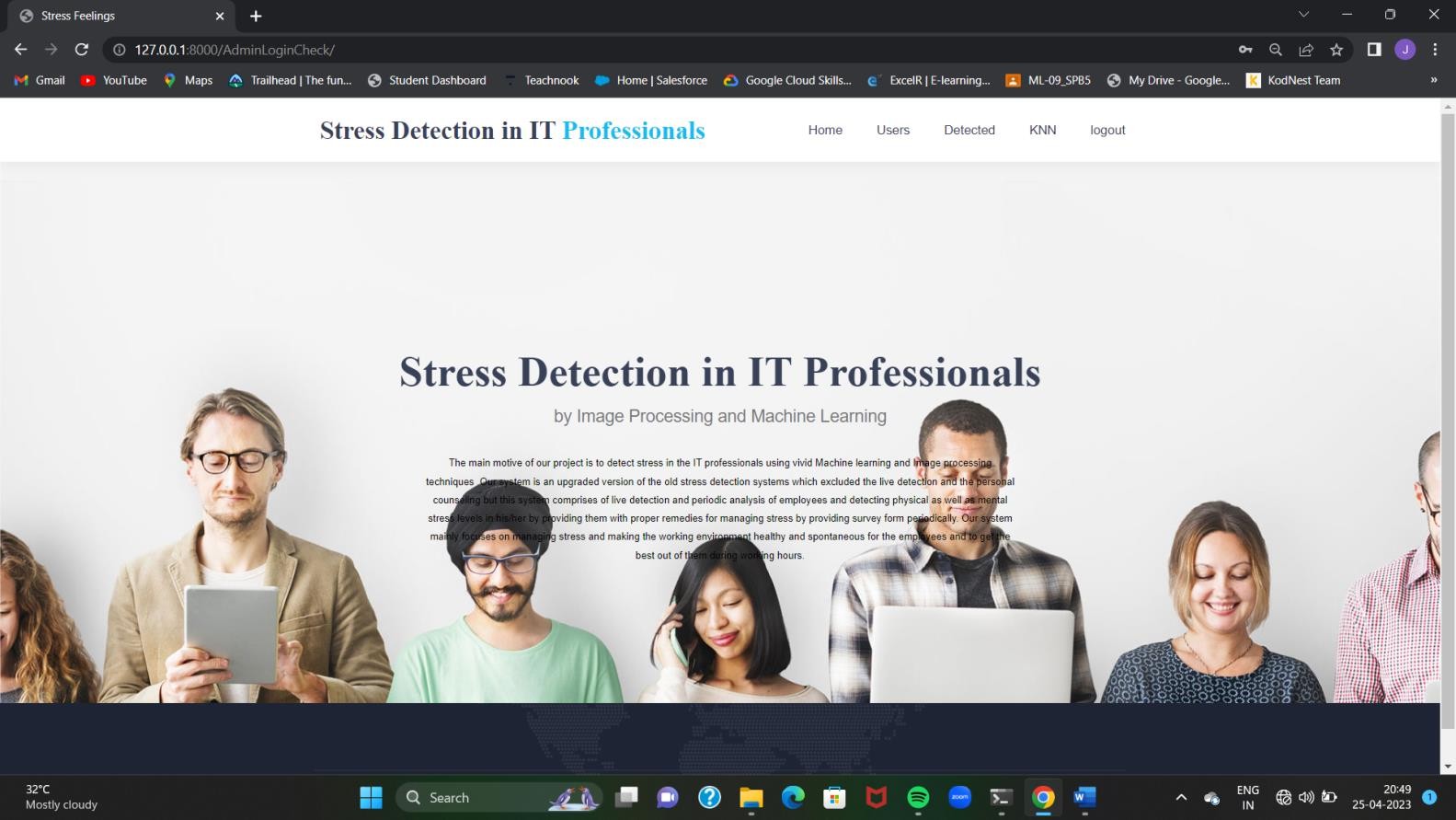
**Live Stream:**

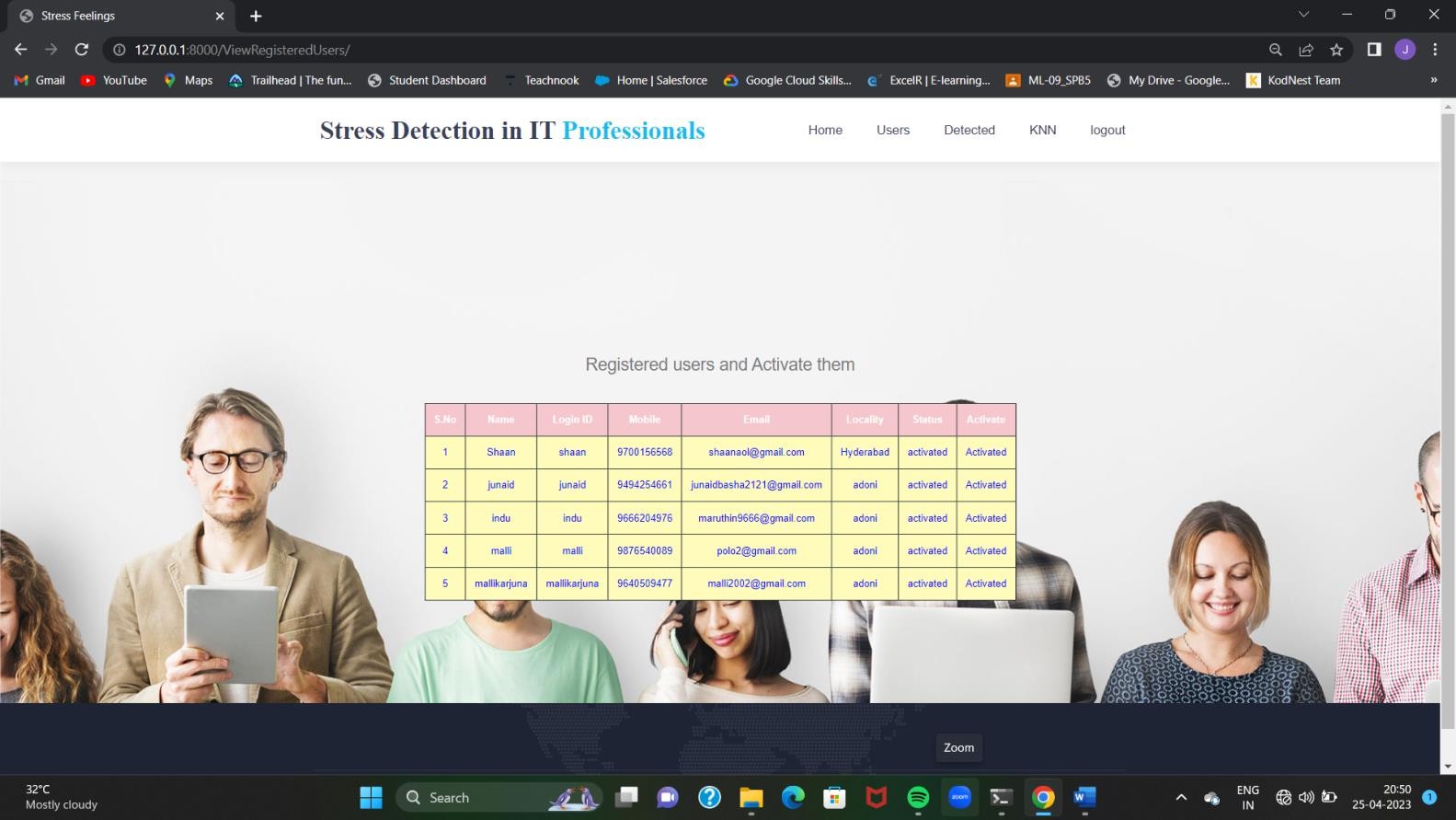
**KNN Results:**

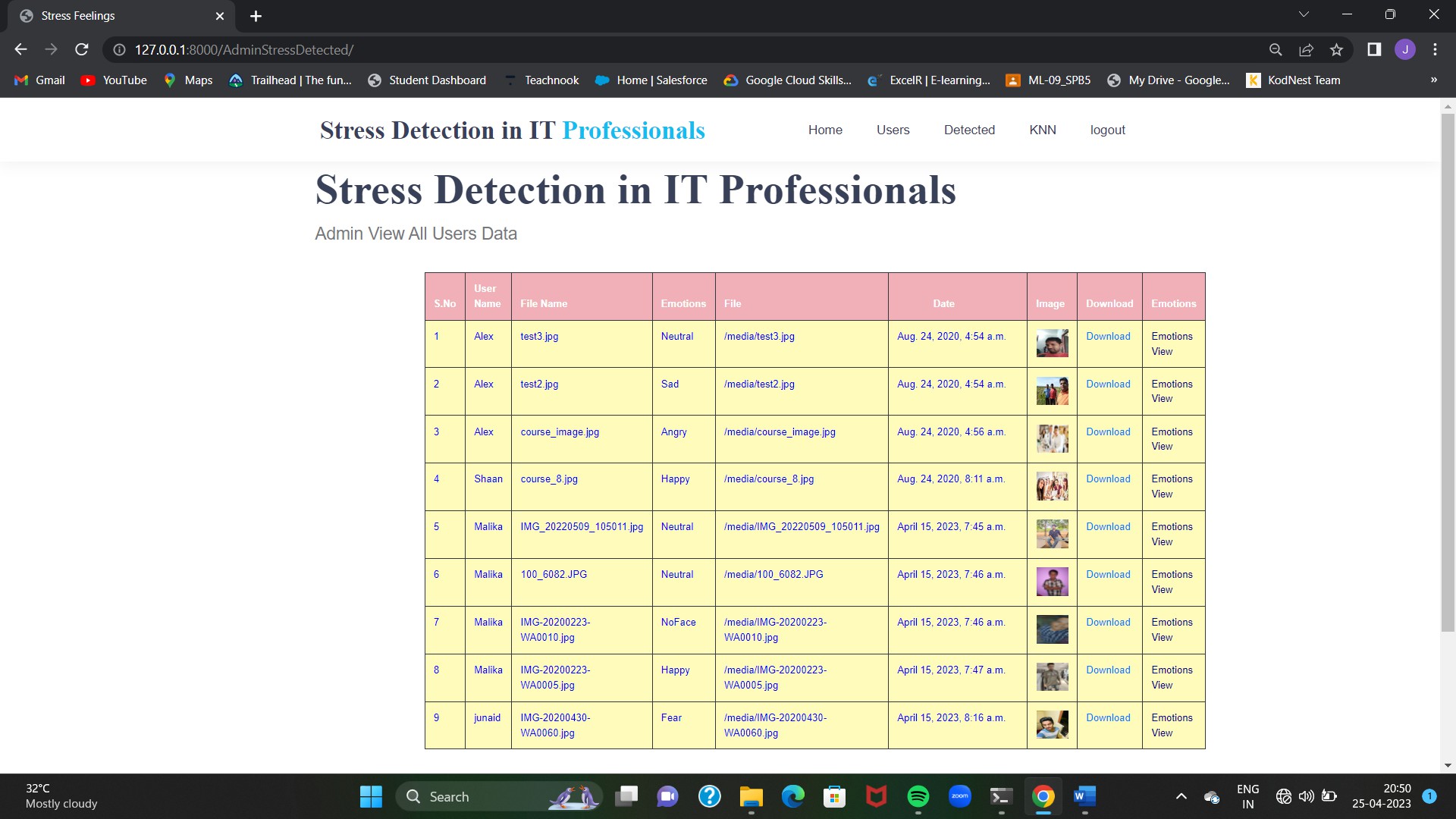
**Dataset View:**

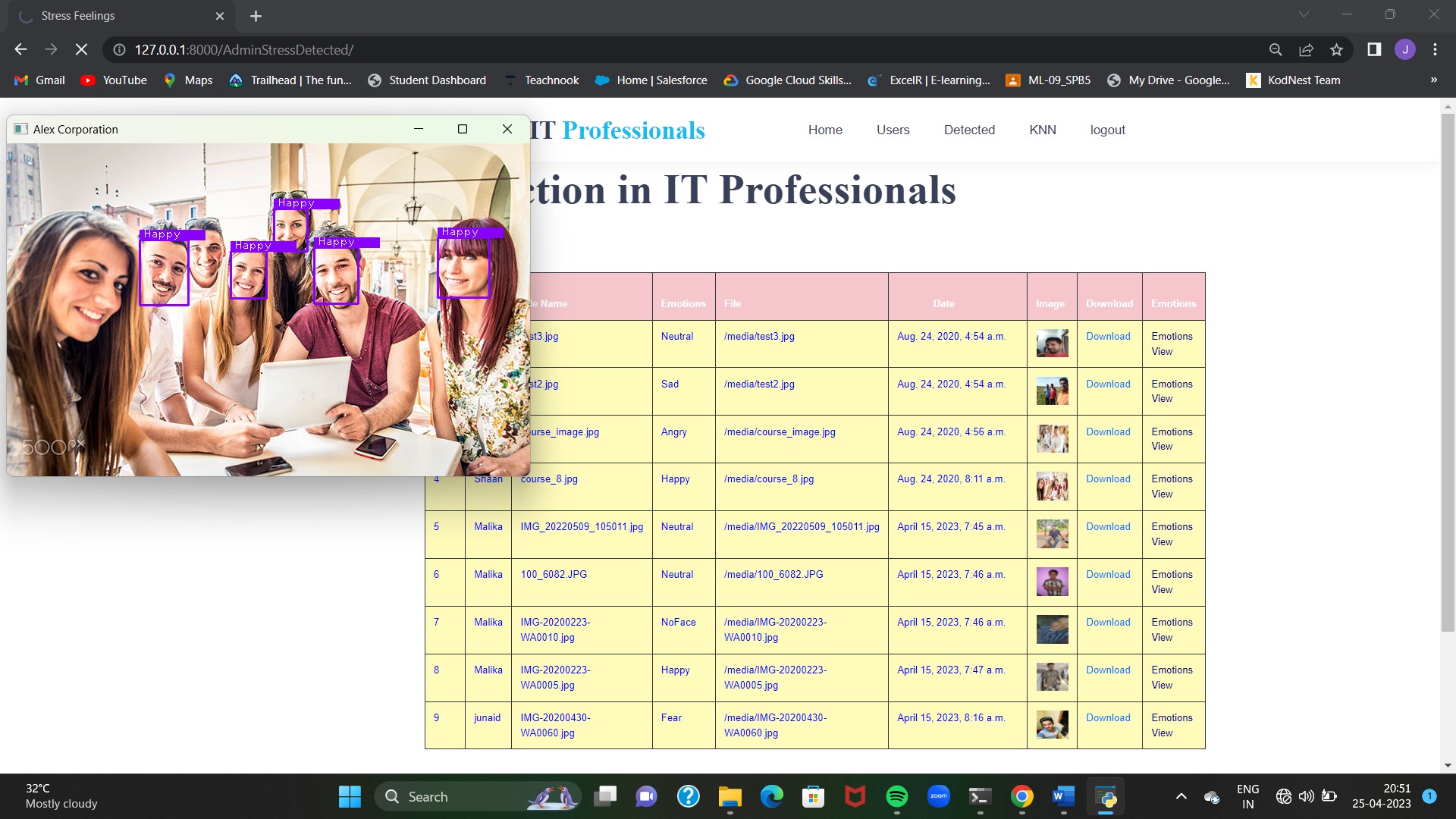
**Admin Page:**

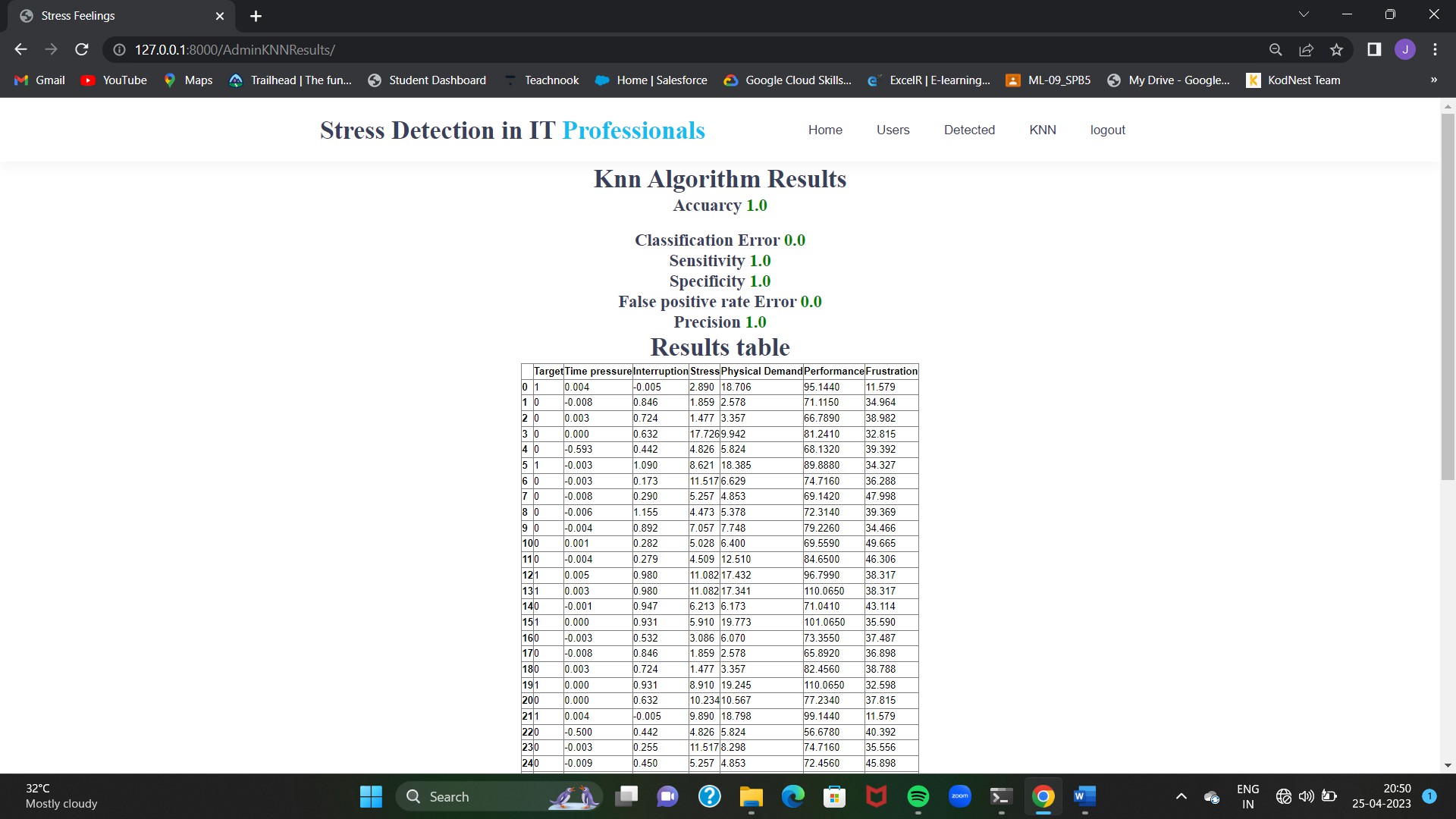
**Admin Home Page:**



**Activate users:**

**Detected Images:**

**Admin Side Resuts:**

**Admin View knn Results:**

# CHAPTER 9 SOFTWARE ENVIRONMENTS:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language) [,](https://en.wikipedia.org/wiki/Interpreted_language)Python has a design philosophy that emphasizes code [readability(](https://en.wikipedia.org/wiki/Readability)notably using [white space](https://en.wikipedia.org/wiki/Whitespace_character) [i](https://en.wikipedia.org/wiki/Whitespace_character)ndentation to delimit [code](https://en.wikipedia.org/wiki/Code_block) [blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) [t](https://en.wikipedia.org/wiki/Source_lines_of_code)han might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B) [o](https://en.wikipedia.org/wiki/C%2B%2B)[r](https://en.wikipedia.org/wiki/Java_(programming_language)) [Java.](https://en.wikipedia.org/wiki/Java_(programming_language))It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system)[.](https://en.wikipedia.org/wiki/CPython) [C Python,](https://en.wikipedia.org/wiki/CPython) the[reference](https://en.wikipedia.org/wiki/Reference_implementation) [implementation](https://en.wikipedia.org/wiki/Reference_implementation) [o](https://en.wikipedia.org/wiki/Reference_implementation)f Python, is [o](https://en.wikipedia.org/wiki/Open_source)pen sourc[e](https://en.wikipedia.org/wiki/Open_source) [s](https://en.wikipedia.org/wiki/Open_source)oftware and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit [Python Software](https://en.wikipedia.org/wiki/Python_Software_Foundation) [Foundation.](https://en.wikipedia.org/wiki/Python_Software_Foundation) Python features a[dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) [s](https://en.wikipedia.org/wiki/Dynamic_type)ystem and automatic [m](https://en.wikipedia.org/wiki/Memory_management)emory managemen[t.](https://en.wikipedia.org/wiki/Memory_management) It supports multiple[programming](https://en.wikipedia.org/wiki/Programming_paradigm) [paradigms,](https://en.wikipedia.org/wiki/Programming_paradigm)including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming)[,imperative](https://en.wikipedia.org/wiki/Imperative_programming)[,functionala](https://en.wikipedia.org/wiki/Functional_programming)nd[procedural,](https://en.wikipedia.org/wiki/Procedural_programming)and has a large and comprehensive [standard library.](https://en.wikipedia.org/wiki/Standard_library)

**Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information. >>> Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");.However in Python version 2.4.3, this produces the following result −

Hello, Python!

**Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo print "Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo #!/usr/bin/python

print "Hello, Python!"

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

$ chmod +x test.py # This is to make file executable $./test.py This produces the following result −

Hello, Python!

**Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

* Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
* Starting an identifier with a single leading underscore indicates that the identifier is private.
* Starting an identifier with two leading underscores indicates a strongly private identifier.
* If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

**Reserved Words**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only. def if return del import try elif in while else is with except lambda yield

### Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

if True:

print "True" else:

print "False"

However, the following block generates an error −

if True:

print "Answer"

print "True" else: print "Answer" print

"False"

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

#!/usr/bin/python

import sys try:

# open file stream file = open(file\_name, "w") except IOError:

print "There was an error writing to", file\_namesys.exit() print "Enter '", file\_finish, print "' When finished" while file\_text != file\_finish:

file\_text = raw\_input("Enter text: ") if file\_text

== file\_finish:

# close the file file.close break file.write(file\_text) file.write("\n") file.close() file\_name = raw\_input("Enter filename: ") if len(file\_name) == 0:

print "Next time please enter something" sys.exit() try:

file = open(file\_name, "r") except IOError:

print "There was an error reading file" sys.exit() file\_text = file.read() file.close() print file\_text

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example − total = item\_one + \ item\_two + \ item\_three

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

word = 'word' sentence = "This is a sentence." paragraph = """This is a paragraph. It is made up of multiple lines and sentences."""

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo #!/usr/bin/python

# First comment

print "Hello, Python!" # second comment This produces the following result −

Hello, Python!

You can type a comment on the same line after a statement or expression −

name = "Madisetti" # This is again comment You can comment multiple lines as follows −

# This is a comment.

# This is a comment, too. # This is a comment, too. # I said that already.

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

'''

This is a multiline comment. '''

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

#!/usr/bin/python

raw\_input("\n\nPress the enter key to exit.")

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application. Multiple Statements on a Single Line

The semicolon ( ; ) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

import sys; x = 'foo'; sys.stdout.write(x + '\n') Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon ( : ) and are followed by one or more lines which make up the suite. For example −

if expression :

suite elifexpression

: suite else

:

suite

**Command Line Arguments**

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

$ python -h usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ... Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

**Python Lists**

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000]; list2

= [1, 2, 3, 4, 5 ]; list3 = ["a",

"b", "c", "d"]

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 ); tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing −

tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

tup1 = (50,);

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo #!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5, 6, 7 ); print "tup1[0]: ", tup1[0]; print "tup2[1:5]: ", tup2[1:5];

When the above code is executed, it produces the following result −

tup1[0]: physics tup2[1:5]: [2, 3,4,5]

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo #!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name'] print "dict['Age']: ", dict['Age']

When the above code is executed, it produces the following result −

dict['Name']: Zara dict['Age']:

7

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo #!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'} print "dict['Alice']: ", dict['Alice']

When the above code is executed, it produces the following result −

dict['Alice']:

Traceback (most recent call last):

File "test.py", line 4, in <module> print "dict['Alice']: ", dict['Alice']; KeyError: 'Alice'

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo #!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'} dict['Age'] = 8; # update existing entry dict['School'] = "DPS School"; # Add new entry print "dict['Age']:

", dict['Age'] print "dict['School']: ", dict['School']

When the above code is executed, it produces the following result −

dict['Age']: 8

dict['School']: DPS School Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo #!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'} del dict['Name']; # remove entry with key 'Name' dict.clear(); # remove all entries in dict del dict ; # delete entire dictionary

print "dict['Age']: ", dict['Age'] print "dict['School']: ", dict['School']

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

dict['Age']:

Traceback (most recent call last): File "test.py", line 8, in <module> print "dict['Age']: ", dict['Age'];

TypeError: 'type' object is unsubscriptable

Note − del() method is discussed in subsequent section.

**Properties of Dictionary Keys**

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

1. More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example

−

Live Demo #!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'} print "dict['Name']: ", dict['Name']

When the above code is executed, it produces the following result −

dict['Name']: Manni

1. Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo #!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7} print "dict['Name']: ", dict['Name']

When the above code is executed, it produces the following result −

Traceback (most recent call last): File "test.py", line 3, in <module>dict =

{['Name']: 'Zara', 'Age': 7}; TypeError: unhashable type: 'list'

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo #!/usr/bin/python

tup1 = (12, 34.56); tup2

= ('abc', 'xyz');

# Following action is not valid for tuples # tup1[0] = 100;

# So let's create a new tuple as follows tup3 = tup1

+ tup2; print tup3;

When the above code is executed, it produces the following result −

(12, 34.56, 'abc', 'xyz')

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo #!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000); print tup; del tup; print "After deleting tup : "; print tup;

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

('physics', 'chemistry', 1997, 2000) After deleting tup :

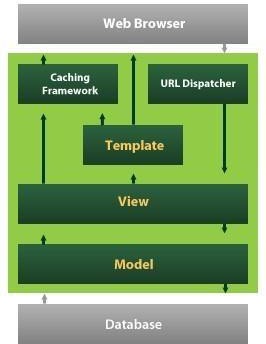
Traceback (most recent call last): File "test.py", line 9, in <module> print tup;

NameError: name 'tup' is not defined

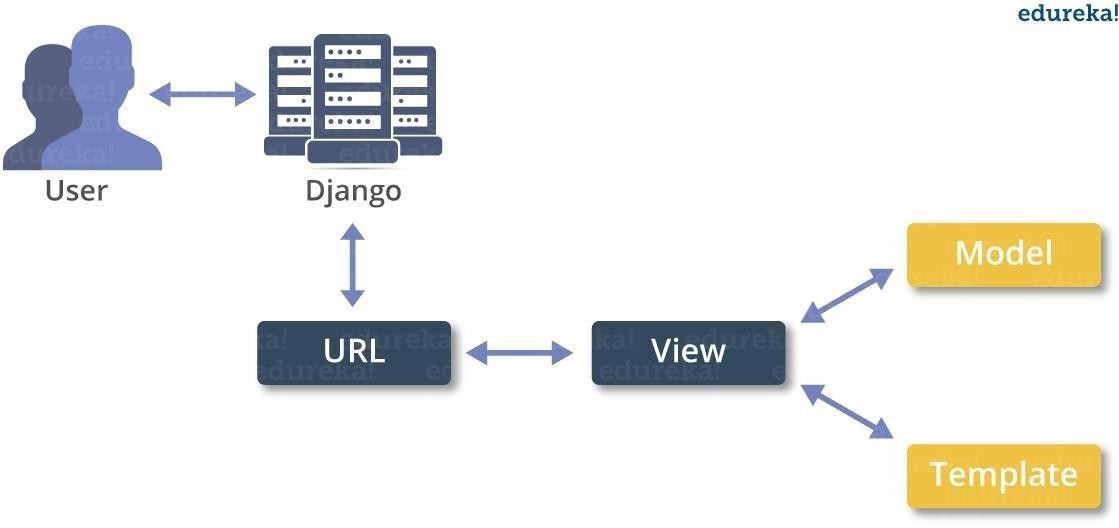
## DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes[reusabilityan](https://en.wikipedia.org/wiki/Reusability)d "pluggability" of components, rapid development, and the principle of[don't](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself) [repeat yourself.](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself)Python is used throughout, even for settings files and data models.



Django also provides an optional administrative[create, read, update and deletei](https://en.wikipedia.org/wiki/Create%2C_read%2C_update_and_delete)nterface that is generated dynamically through[introspectiona](https://en.wikipedia.org/wiki/Introspection_(computer_science))nd configured via admin models



**Create a Project**

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin startprojectmyproject

This will create a "myproject" folder with the following structure −

myproject/ manage.py myproject/

init .py

settings.py urls.py wsgi.py

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements − manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

init .py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project.

However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3', 'NAME': 'database.sql',

'USER': '',

'PASSWORD': '',

'HOST': '',

'PORT': '',

}

}

Database is set in the ‘Database’ dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)

PostGreSQL (django.db.backends.postgresql\_psycopg2) Oracle (django.db.backends.oracle) and NoSQL DB MongoDB (django\_mongodb\_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME\_ZONE, LANGUAGE\_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

$ python manage.py runserver

You will get something like the following on running the above code −

Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings' Starting development server at http://127.0.0.1:8000/ Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

**Create an Application**

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

$ python manage.py startappmyapp

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

myapp/

init .py admin.py

models.py tests.py views.py

init .py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED\_APPS tuple in the settings.py file of your project (add your app name) −

INSTALLED\_APPS = (

'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'myapp',

)

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

#-\*- coding: utf-8 -\*- from django import forms

class LoginForm(forms.Form):

user = forms.CharField(max\_length = 100)

password = forms.CharField(widget = forms.PasswordInput())

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django:

DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

#-\*- coding: utf-8 -\*- from myapp.forms import LoginForm def login(request):

username = "not logged in"

if request.method == "POST": #Get the posted form

MyLoginForm = LoginForm(request.POST)

if MyLoginForm.is\_valid():

username = MyLoginForm.cleaned\_data['username'] else: MyLoginForm = Loginform()

return render(request, 'loggedin.html', {"username" : username})

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

<html>

<body>

<form name = "form" action = "{% url "myapp.views.login" %}" method = "POST" >{% csrf\_token %}

<div style = "max-width:470px;">

<center>

<input type = "text" style = "margin-left:20%;" placeholder = "Identifiant" name

= "username" /></center>

</div>

<br>

<div style = "max-width:470px;">

<center>

<input type = "password" style = "margin-left:20%;" placeholder = "password" name = "password" />

</center>

</div>

<br>

<div style = "max-width:470px;">

<center>

<button style = "border:0px; background-color:#4285F4; margin-top:8%; height:35px; width:80%;margin-left:19%;" type = "submit" value = "Login" ><strong>Login</strong>

</button>

</center>

</div>

</form>

</body>

</html>

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

{% csrf\_token %}

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

<html>

<body>

You are :<strong>{{username}}</strong>

</body>

</html>

Now, we just need our pair of URLs to get started: myapp/urls.py

from django.conf.urls import patterns, url from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

url(r'^connection/',TemplateView.as\_view(template\_name = 'login.html')), url(r'^login/', 'login', name = 'login'))

When accessing "/myapp/connection", we will get the following login.html template rendered − Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE\_CLASSES and the INSTALLED\_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE\_CLASSES should have −

'django.contrib.sessions.middleware.SessionMiddleware' And INSTALLED\_APPS should have −

'django.contrib.sessions'

By default, Django saves session information in database (django\_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won’t see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side −

def login(request): username = 'not logged in'

if request.method == 'POST':

MyLoginForm = LoginForm(request.POST)

if MyLoginForm.is\_valid():

username = MyLoginForm.cleaned\_data['username'] request.session['username'] = username else:

MyLoginForm = LoginForm()

return render(request, 'loggedin.html', {"username" : username}

Then let us create formView view for the login form, where we won’t display the form if cookie is set −

def formView(request):

if request.session.has\_key('username'): username = request.session['username'] return render(request, 'loggedin.html',

{"username" : username}) else:

return render(request, 'login.html', {})

Now let us change the url.py file to change the url so it pairs with our new view −

from django.conf.urls import patterns, url from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views', url(r'^connection/','formView', name = 'loginform'), url(r'^login/',

'login', name = 'login'))

When accessing /myapp/connection, you will get to see the following page

# CHAPTER 10 CONCLUSION

Stress Detection System is designed to predict stress in the employees by monitoring captured images of authenticated users which makes the system secure. The image capturing is done automatically when the authenticate user is logged in based on some time interval. The captured images are used to detect the stress of the user based on some standard conversion and image processing mechanisms. Then the system will analyze the stress levels by using Machine Learning algorithms which generates the results that are more efficient.

CHAPTER 11 REFERENCES

1. G. Giannakakis, D. Manousos, F. Chiarugi, “Stress and anxiety detection using

facial cues from videos,” Biomedical Signal processing and Control”, vol. 31, pp. 89-101, January 2017.

1. T. Jick and R. Payne, “Stress at work,” Journal of Management Education, vol. 5, no. 3, pp. 50- 56, 1980.
2. Nisha Raichur, Nidhi Lonakadi, Priyanka Mural, “Detection of Stress Using Image Processing and Machine Learning Techniques”, vol.9, no. 3S, July 2017.
3. Bhattacharyya, R., &Basu, S. (2018). Retrieved from ‘The Economic Times’.
4. OSMI Mental Health in Tech Survey Dataset, 2017
5. U. S. Reddy, A. V. Thota and A. Dharun, "Machine Learning Techniques for Stress Prediction in Working Employees," 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Madurai, India, 2018, pp. 1-4.
6. https:/[/www.kaggle.com/qiriro/stress](http://www.kaggle.com/qiriro/stress)
7. Communications, N.. World health report. 2001. URL: [http://www.who.int/whr/2001/media\_centre/press\_release/en/.](http://www.who.int/whr/2001/media_centre/press_release/en/)
8. Bakker, J., Holenderski, L., Kocielnik, R., Pechenizkiy, M., Sidorova, N..Stess@ work: From measuring stress to its understanding, prediction and handling with personalized coaching. In: Proceedings of the 2nd ACM SIGHIT International health informatics symposium. ACM; 2012,

p. 673–678.

1. Deng, Y., Wu, Z., Chu, C.H., Zhang, Q., Hsu, D.F.. Sensor feature selection and combination for stress identification using combinatorial fusion. International Journal of Advanced Robotic Systems 2013;10(8):306.
2. Ghaderi, A., Frounchi, J., Farnam, A.. Machine learning-based signal processing using physiological signals for stress detection. In: 2015 22nd Iranian Conference on Biomedical Engineering (ICBME). 2015, p. 93–98.
3. Villarejo, M.V., Zapirain, B.G., Zorrilla, A.M.. A stress sensor based on galvanic skin response (gsr) controlled by zigbee. Sensors 2012; 12(5):6075–6101.
4. Liu, D., Ulrich, M.. Listen to your heart: Stress prediction using consumer heart rate sensors 2015;.
5. Nakashima, Y., Kim, J., Flutura, S., Seiderer, A., Andre, E.. Stress recognition in daily work. In:

´ International Symposium on Pervasive Computing Paradigms for Mental Health. Springer; 2015, p. 23–33.

1. Xu, Q., Nwe, T.L., Guan, C.. Cluster-based analysis for personalized stress evaluation using physiological signals. IEEE journal of biomedical and health informatics 2015;19(1):275–281.
2. Tanev, G., Saadi, D.B., Hoppe, K., Sorensen, H.B.. Classification of acute stress using linear and non-linear heart rate variability analysis derived from sternal ecg. In: Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE. IEEE; 2014, p. 3386–3389.
3. Gjoreski, M., Gjoreski, H., Lustrek, M., Gams, M.. Continuous stress detection using a wrist device: in laboratory and real life. In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct.

ACM; 2016, p. 1185–1193.

1. Palanisamy, K., Murugappan, M., Yaacob, S.. Multiple physiological signal-based human stress identification using non-linear classifiers.Elektronikairelektrotechnika

2013;19(7):80–85.

1. Widanti, N., Sumanto, B., Rosa, P., Miftahudin, M.F.. Stress level detection using heart rate, blood pressure, and gsr and stress therapy by utilizing infrared. In: Industrial Instrumentation and Control (ICIC), 2015 International Conference on. IEEE; 2015, p. 275– 279.
2. Sioni, R., Chittaro, L.. Stress detection using physiological sensors. Computer 2015;48(10):26– 33.
3. Zenonos, A., Khan, A., Kalogridis, G., Vatsikas, S., Lewis, T., Sooriyabandara, M..Healthyoffice: Mood recognition at work using smartphones and wearable sensors. In: Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on. IEEE; 2016, p. 1–6.
4. Selvaraj, N.. Psychological acute stress measurement using a wireless adhesive biosensor. In: 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). 2015, p. 3137–3140.
5. Vadana, D.P., Kottayil, S.K.. Energy-aware intelligent controller for dynamic energy management on smart microgrid. In: Power and Energy Systems Conference: Towards Sustainable Energy, 2014. IEEE; 2014, p. 1–7.
6. Rajagopalan, S.S., Murthy, O.R., Goecke, R., Rozga, A.. Play with memeasuring a

child’s engagement in a social interaction. In: Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on; vol. 1. IEEE; 2015, p.

1–8.

1. Koldijk, S., Neerincx, M.A., Kraaij, W.. Detecting work stress in offices by combining unobtrusive sensors. IEEE Transactions on Affective Computing 2016;.
2. Koldijk, S., Sappelli, M., Verberne, S., Neerincx, M.A., Kraaij, W.. The swell knowledge work dataset for stress and user modeling research. In: Proceedings of the 16th International Conference on Multimodal Interaction; ICMI ’14. New York, NY, USA: ACM. ISBN 978-1- 4503-2885-2; 2014, p. 291–298. URL: [http://doi.acm.org/10.1145/2663204.2663257.](http://doi.acm.org/10.1145/2663204.2663257)

doi:10.1145/2663204.2663257. [27]Yoder, N..Peakfinder. Internet: [http://www](http://www/) mathworks com- /matlabcentral/fileexchange/25500 2011;.