CHAPTER-1
INTRODUCTION

1.1 INTRODUCTION

A specific strain exerted on the human body as a result of a number of stimuli is also known as stress. The human body releases stress hormones when it is under stress. Absolute, physiological, relative stressors, and psychological are all different types of stressors [1]. Additionally, stress not only affects your attitude, your relationships, energy level, and duty performance, but it can also cause or aggravate a variety of medical conditions. Therefore, sleep is an essential component in preserving human homeostasis. Sleep disturbances are related to a number of physical, mental, and social problems. In many nations, chronic sleep deprivation is becoming a common. Because the body's stress systems are so essential in adjusting to a continuously moving and stressful background, it's crucial to understand how sleep loss affects them. The human body activates its defenses systems in an adaptive effort to maintain homeostasis. Insomnia may happen if these protections are ineffective. A disruption in routine, such as a psychiatric condition, an illness, or worry, can result in short-term insomnia [2]. The major goal of this study is to detect how human stress change based on sleeping habits. Also the specific objectives are identify the benefits of using a model to detect human stress, identify the relationship between human stress and sleeping habits, identify the major human sleeping behaviors 979-8-3503-4737-1/23/$31.00 ©2023 IEEE R.A.H.M. Rupasingha Department of Economics and Statistics Sabaragamuwa University of Sri Lanka Belihuloya, Sri Lanka hmrupasingha@gmail.com that affect a person's stress, identify the techniques we can use for detecting stress of human and finally, detecting the stress of human based on sleeping habits. We used data mining methods to build a model to determine the level of stress of the people before and after sleep. It is a tool that enables the study, examination, and visualization of extremely large data sets at a high level of abstraction in data mining. Six different data mining techniques, including Decision Trees, Naïve Bayes, SVM, Random Forest, MLP, and Logistic Regression, are utilized to categorize stress levels. Data can be analyzed by these ML algorithms to predict whether or not people would feel worried the next day. Here, we can assess human's sleep habits and degrees of stress while teaching them how to find human stress before, after and during sleep. The bulk of available studies predicts human stress in and through sleep using just a few independent variables. However, it is difficult to forecast human stress when there are so few independent variables. By extending the number of attributes in our study, we were able to consider many more. respiration rate, Snoring range, limb movement rate, body temperature, eye movement, blood oxygen level, sleep time, heart rate, and stress levels are just few of the factors to consider. Also we increased the number of classification models to overcome the limitations of existing approaches. This study looks into several taxonomic techniques for predicting human stress in and during. And the main objective of this study is to detect the way of human stress change based on sleeping habits. In this study, successfully outperform Decision Tree (J48), Random Forest, SVM, MLP and Logistic Regression results by employing a Naive Bayes machine-learning algorithm. We collect information on human behaviors based on the categories listed above. The optimal method among these six algorithms for the suggested strategy was identified based on accuracy, precision, f-measure, recall, RMSE and MAE values. Then take the 10-folds cross validation with the maximum accuracy by using the WEKA data mining tool. The paper's reminding is structured as follows. We go over similar work in section II. The methodology of the research is discussed in Section III. The Section IV is discussed evaluation and experiments, and Section V discusses the future work..

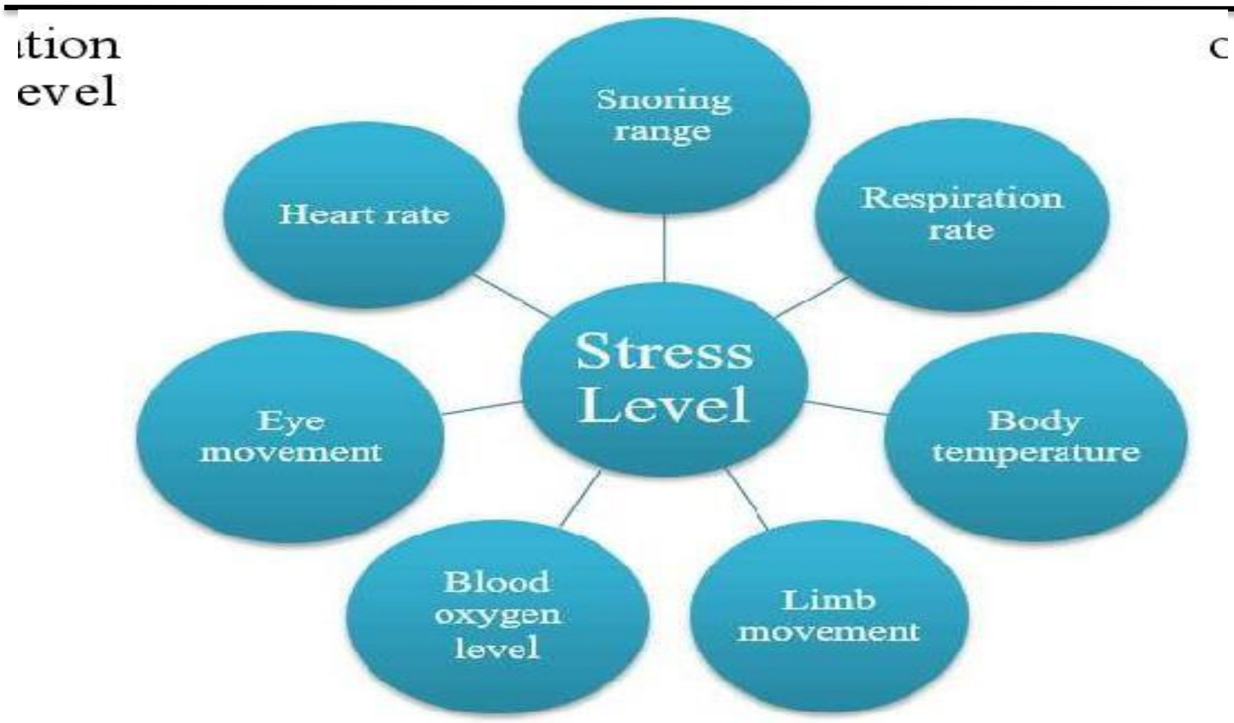


Fig 1: Stress levels

CHAPTER- 2
LITERATURE SURVEY

2.1 LITERATURE REVIEW

A Comprehensive Literature Survey on Stress Detection and Management Through Sleeping Habits Using Machine Learning

Abstract:

Stress is a significant factor influencing mental and physical well-being, and its relationship with sleep has been extensively studied. This literature survey explores existing research on stress and sleep, highlighting various psychological, behavioral, and biological determinants. The survey includes foundational studies on stress and health ([1]), the impact of sleep disorders on stress ([2]), and learning stress among students ([3]). Further discussions include cognitive and emotional perspectives on stress ([4]), stress assessment techniques ([5]), machine learning-based personalized sleep estimation ([6]), and blockchain-integrated frameworks for stress management ([7]). This review provides a detailed exploration of existing methodologies and technological advancements in stress detection, emphasizing machine learning and privacy-preserving frameworks.

Literature Review:

1. Stress and Health: Psychological, Behavioral, and Biological Determinants ([1])

- This study examines how stress affects human health through psychological, behavioral, and biological factors.
- It discusses how chronic stress can lead to cardiovascular diseases, immune dysfunction, and mental health disorders.
- The research highlights the need for better stress detection and management techniques.

2. Sleep, Sleep Disorders, and Stress ([2])

- Explores the bidirectional relationship between sleep and stress.
- Sleep deprivation increases stress levels, while chronic stress contributes to sleep disorders such as insomnia.
- The study suggests that proper sleep hygiene and interventions can reduce stress.

3. Learning Stress and Stress Management Among Postgraduate Students ([3])

- Investigates the sources of academic stress and strategies for managing it among postgraduate students.
- The study finds that exam pressure, workload, and future career concerns contribute to stress.
- Coping mechanisms such as relaxation techniques and counseling are discussed.

4. Stress: Concepts, Cognition, Emotion, and Behavior ([4])

- Provides an overview of stress from a psychological and cognitive perspective.
- Discusses how emotions and thought processes contribute to stress.
- The study highlights the importance of early stress detection to prevent long-term health consequences.

5. Stress Assessment and Development of a Primary Care Psychological Service ([5])

- Focuses on different methodologies for stress assessment.
- Proposes a structured psychological care model for stress management.
- Emphasizes the role of mental health professionals in stress assessment and treatment.

6. Personalized Sleep Parameters Estimation from Actigraphy Using Machine Learning ([6])

- Utilizes machine learning to analyze actigraphy data for personalized sleep estimation.
- The study highlights how sleep patterns can be used to predict stress levels.
- Machine learning models improve the accuracy of sleep monitoring and stress detection.

7. SaYoPillow: Blockchain-Integrated IoMT Framework for Stress Management ([7])

- Proposes a blockchain-based Internet of Medical Things (IoMT) framework for stress management.
- Uses sleeping habits as a key factor in stress assessment.
- Ensures privacy and security in stress monitoring through blockchain technology.

CHAPTER-3 SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

The existing system for stress detection primarily relies on traditional methods such as self-reported questionnaires, clinical assessments, and basic wearable devices that monitor heart rate and sleep duration. These methods, while useful, have several limitations. One major drawback of self-reported stress assessments is their subjectivity—individuals may underreport or over report their stress levels based on their perception rather than actual physiological data. Additionally, these assessments do not provide real-time or continuous monitoring, making them ineffective for early intervention. Clinical methods, on the other hand, require professional supervision and can be time-consuming and costly. Furthermore, existing wearable devices that track sleep patterns often lack comprehensive physiological metrics, limiting their accuracy in detecting stress. They usually focus on heart rate and sleep duration but fail to consider other crucial physiological indicators such as body temperature, respiration rate, blood oxygen levels, and limb movement, which can provide a more holistic understanding of stress. Additionally, traditional methods lack advanced machine learning algorithms to analyze complex patterns in sleep data, making their stress classification accuracy relatively low.

3.2 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

  .

3.3 PROPOSED SYSTEM:

- The proposed system aims to address the limitations of traditional stress detection methods by leveraging machine learning algorithms for a more accurate and automated analysis of stress levels based on sleep patterns. It utilizes a dataset containing multiple sleep-related features such as snoring range, respiration rate, body temperature, limb movement, blood oxygen levels, eye movement, sleep hours, heart rate, and stress levels.
- By implementing multiple machines learning models, including Random Forest, Support Vector Machine (SVM), Decision Trees (DT), Naïve Bayes (NB), and Logistic Regression (LR), the system ensures a comparative performance analysis to determine the most effective model for stress classification. Machine learning allows the system to identify hidden patterns and correlations in sleep behavior that may not be easily detectable through traditional methods.
- Another significant advantage of the proposed system is its automated and real-time stress prediction capability. By integrating multiple physiological indicators, the system provides a more accurate, data-driven approach to stress detection. It also reduces reliance on subjective self-reporting, ensuring that stress assessment is based on quantifiable physiological data rather than perception alone.
- Additionally, the proposed system can be integrated into wearable health devices, enabling continuous monitoring and early detection of stress-related issues. This not only helps individuals manage stress proactively but also assists healthcare professionals in personalizing stress management interventions based on real-time physiological insights.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM:

- **Early Stress Detection:** Identifying stress at an early stage can help prevent severe mental and physical health issues.
- **Non-Intrusive Monitoring:** Unlike traditional methods that require active participation,

this system passively analyzes sleep data to determine stress levels.

- **Improved Health and Productivity:** Reducing stress can lead to better sleep quality, increased focus, and enhanced overall well-being.
- **Low Complexities and User-Friendly**
- **Contribution to Mental Health Research:** This project provides valuable insights into the correlation between sleep behavior and stress, supporting further research in mental health and wellness.

CHAPTER-4
SYSTEM REQUIREMENTS SPECIFICATION

6. SYSTEM REQUIREMENTS

Requirement Specification, also known as Documentation, is a procedure of noting all the system and user requirements in the form of a document. These requirements must be clear, comprehensive, complete and consistent. During the requirement gathering, we have to gather all the specifications from different sources.

6.1.1 HARDWARE REQUIREMENTS:

• System	: Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz
• Hard Disk	: 1 TB.
• Input Devices	: Keyboard, Mouse
• Ram	: 4 GB.

6.1.2 SOFTWARE REQUIREMENTS:

➢ Operating system	:	Windows 10
➢ Server-side Script	:	Python 3.9
➢ Tool	:	Anaconda
➢ Framework	:	Flask
➢ Interface	:	Jupiter notebook

CHAPTER- 5

SYSTEM DESIGN

5.1 INTRODUCTION

System Design Introduction:

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

5.2 MODULES

5.2.1 Feature Selection:

The dataset used in this system consists of multiple physiological and behavioral features essential for stress detection. The dataset includes the following features:

- **SR (Snoring Range):** Indicates the intensity and frequency of snoring during sleep.
- **RR (Respiration Rate):** The number of breaths per minute, which can be affected by stress levels.
- **T (Body Temperature):** Variations in body temperature can indicate stress-induced physiological changes.
- **LM (Limb Movement):** Measures physical movement during sleep, as stress can lead to restlessness.
- **BO (Blood Oxygen Level):** Oxygen saturation levels in the blood, which can fluctuate under stress conditions.
- **REM (Rapid Eye Movement):** The duration and frequency of REM sleep, which is crucial for stress and emotional regulation.
- **HR (Heart Rate):** Heart rate fluctuations, which increase with stress and anxiety.
- **SL (Stress Level):** The target variable indicating whether an individual is experiencing stress.

The dataset is preprocessed before feeding it into the Machine learning model for training.

5.2.2 PRE-PROCESSING:

Data preprocessing is a crucial step to ensure that the dataset is clean and structured for accurate model training. The preprocessing phase involves the following steps:

- **Handling Missing Values:** Any missing or incomplete data points are either removed or imputed using statistical methods such as mean, median, or mode.
- **Normalization and Scaling:** Features like heart rate and respiration rate may have varying ranges, so they are normalized to bring them to a common scale. Standardization techniques such as Min-Max scaling are applied.
- **Encoding Categorical Variables:** If the dataset contains categorical variables, they are converted into numerical values using encoding techniques like one-hot encoding.
- **Splitting Dataset:** The dataset is divided into training, validation, and testing sets to ensure proper evaluation of the model's performance.

5.2.3 MACHINE LEANRING ALGORIHTM:

The various algorithms used to measure the level of stress are Random Forest, Gradient Boost, Support Vector Machine, Decision Tree, Naïve Bayes, K-Nearest Neighbours. Among these Random Forest provides higher rate of accuracy i.e 95% as compared to the remaining algorithms

Random Forest Algorithm:

It is a supervised machine learning algorithm. It is made up of many decision trees at every instance it takes a route to generate a valid output. The Random Forest algorithm can be used for each classification resembles the frequency of repetition as well as regression problems results with the mean of the data.
The larger number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. It is also preferable to take small amount of data for training as compared to other algorithms. Even if there is a loss of data in large amount still it gives high accuracy by running efficiently. As the target variable is mentioned in the dataset

and it gets mould itself to adopt changes by the model and predicts the output as per the behaviour of the data.

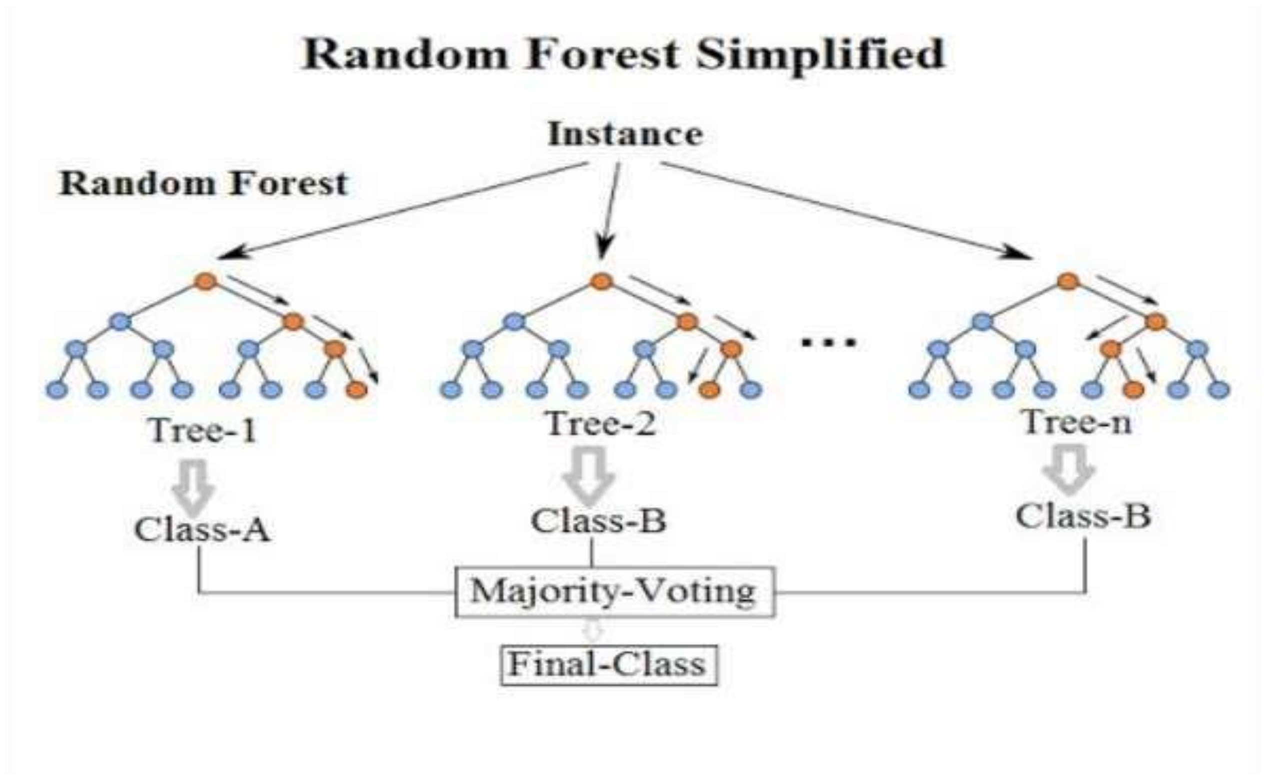


Fig : Random Forest

User Interface :

1. In this project render is used to make the HTML, CSS and JavaScript files to create a use interaction, based environment by using the unique link anyone can be detected their levels of stress. The render environment provides web services to deploy their model and designing the input platform.
2. It is a user-friendly environment to make dynamic changes in the render. The form validation is incorporated within the html file to avoid unnecessary inputs.

Model Accuracy Calculation

After training, the model is evaluated using various performance metrics:

- **Accuracy Score:** The percentage of correctly predicted stress levels.
- **Confusion Matrix:** A breakdown of true positives, true negatives, false positives, and false negatives.
- **Precision, Recall, and F1-score:** To measure the effectiveness of the model in predicting stress levels accurately.

5.2.4 **Predicting Stress Levels**

When a user uploads new sleep-related data, the trained model predicts the **stress level** based on input features. The model processes the data and provides an output indicating whether the user is experiencing **low, moderate, or high stress**.

User Module:

1. User Registration

To use the system, users must first register on the Flask web application. The registration process involves:

- Entering basic details such as name, email, and password.

- Secure password storage using **hashing techniques** like bcrypt to protect user credentials.
- Storing user details in a database for authentication.

2. User Login

Once registered, users can log in to access the system. The login module performs:

- **Authentication:** Checking whether the entered email and password match stored records.
- **Session Management:** Establishing a secure session to track user activity.
- **Error Handling:** Providing alerts for incorrect credentials.

3. Uploading Features to the Flask Web Application

Users are provided with a web interface where they can upload sleep-related features, including:

- Snoring range
- Respiration rate
- Body temperature
- Limb movement
- Blood oxygen levels
- Rapid eye movement
- Sleep hours
- Heart rate

This data can be manually entered or collected from **wearable devices/sensors** and uploaded as a CSV file.

4. Predicting Stress Levels and Displaying Results

Once the user uploads sleep data, the system:

1. **Processes the Data:** Prepares the input in the required format for the trained model.
2.
3. **Generates Predictions:** The model classifies the stress level as **low, moderate, or high**.
4. **Displays Results:** The predicted stress level is shown on the **web dashboard**, along with possible recommendations for stress management.

5. Web-Based Visualization and User Interaction

To improve usability, the web application includes:

- **Graphical Representations:** Displaying trends of stress levels over time.
- **Interactive UI:** A user-friendly interface to input data and receive results.
- **Real-time Insights:** Users can track their stress levels over different periods.

5.3 SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system. Organized in a way that supports reasoning about the structures and behaviors of the system.

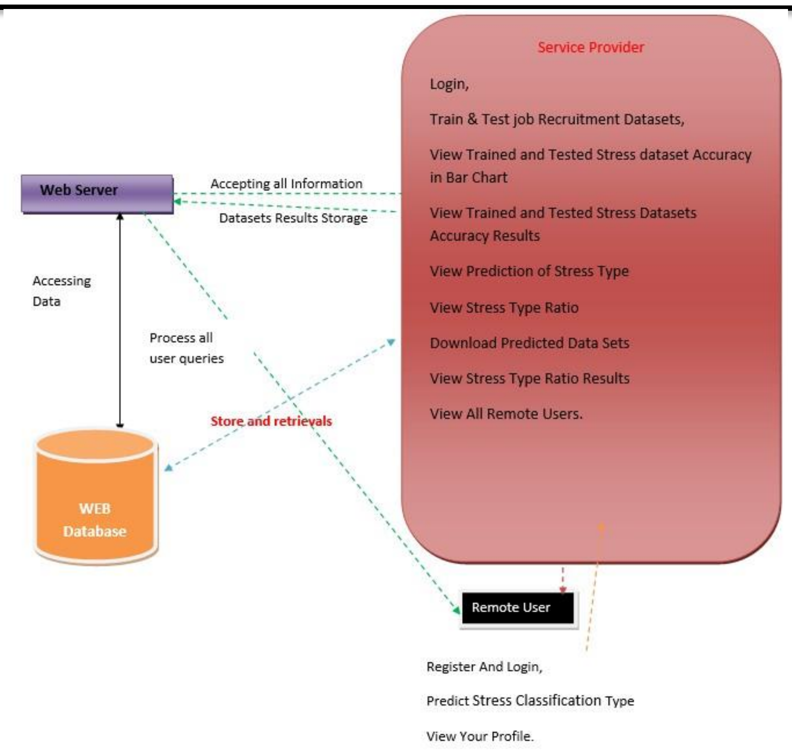

Service Provider
Login,
Train & Test job Recruitment Datasets,
View Trained and Tested Stress dataset Accuracy in Bar Chart
View Trained and Tested Stress Datasets Accuracy Results
View Prediction of Stress Type
View Stress Type Ratio
Download Predicted Data Sets
View Stress Type Ratio Results
View All Remote Users.
Web Server
Accepting all Information
Datasets Results Storage
Accessing Data
Process all user queries
Store and retrievals
WEB Database
Remote User
Register And Login,
Predict Stress Classification Type
View Your Profile.

Figure 5. 1 System Architecture

3-Tier Architecture:

The three-tier software architecture (a three-layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

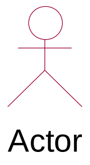
The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

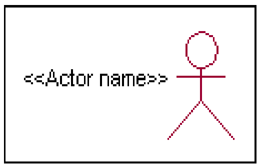
Advantages of Three-Tier:

- Separates functionality from presentation.
- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.
- Effective network performance.

5.4 UML DAIGRAMS

A Use Case Diagram is a type of Unified Modelling Language (UML) diagram one of behavioral diagram defined and created from Use-Case analysis. It represents the interaction between actors(users or external systems),system specific goals and any dependencies between those use cases. It depicts the high-level functionality of system and also informs how the user handles a system.





T

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

- Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

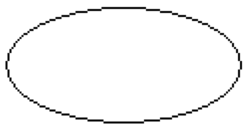
The actors identified in this system are:

a. System Administrator
b. Customer
c. Customer Care

Identification of use cases:

Use case: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?

- Does any actor need to inform about certain occurrences in the system?
- What usecases will support and maintains the system?

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

5.4.1 CONSTRUCTION OF USE CASE DIAGRAMS:

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.

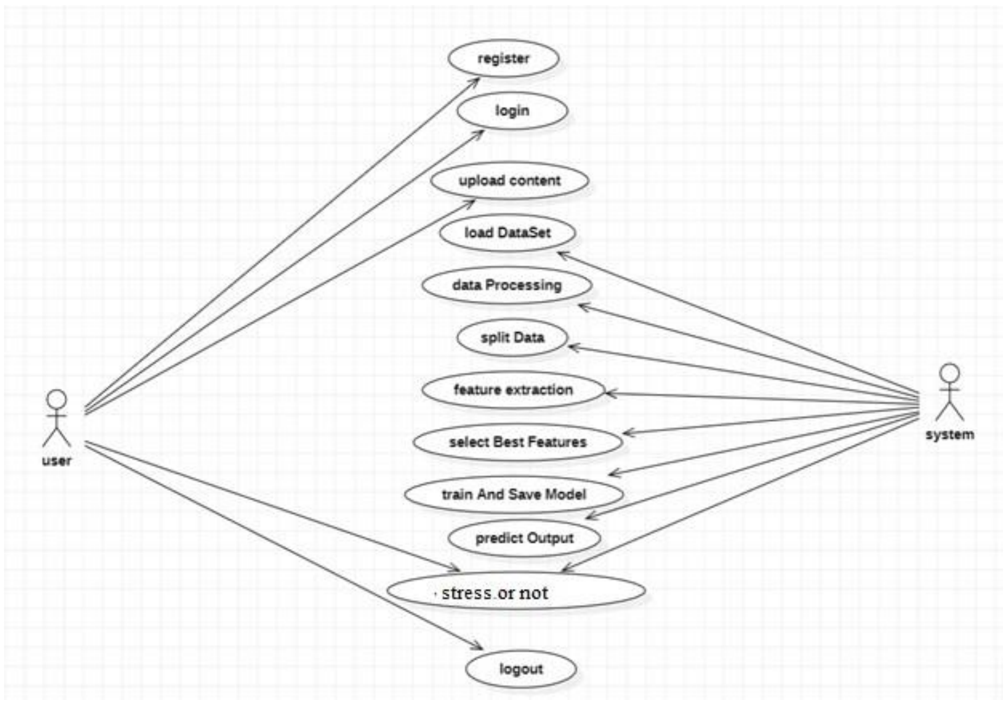


Figure 5. 2 Use Case Diagram

5.4.2 SEQUENCE DIAGRAMS:

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.

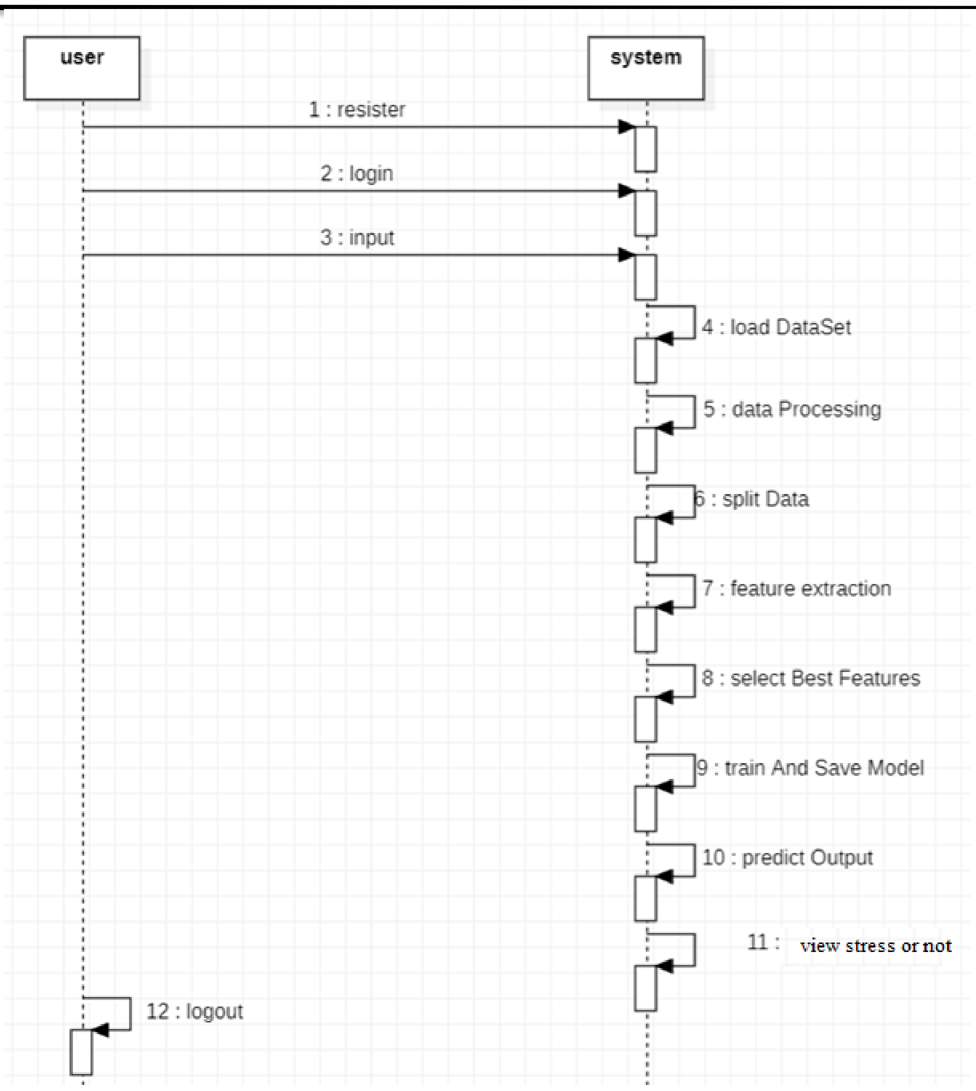


Figure 5. 3 Sequence diagram

5.4.3 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.

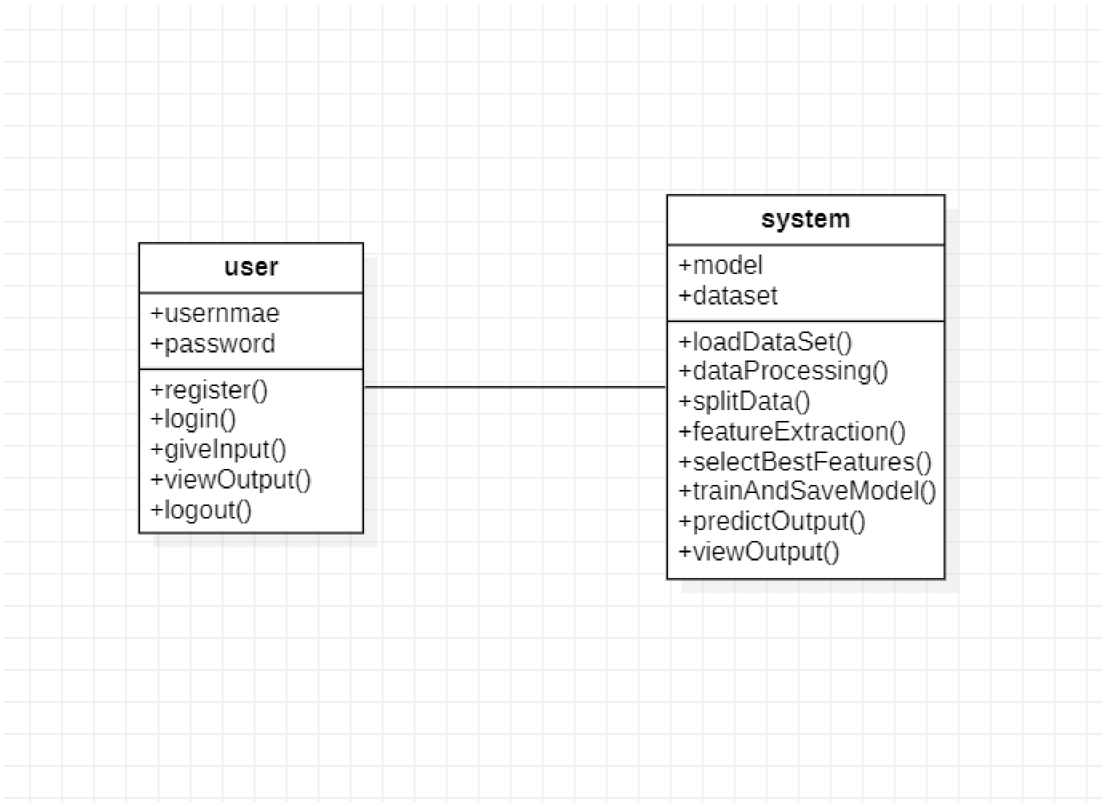


Figure 5. 4 Class Diagram

5.4.4 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-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

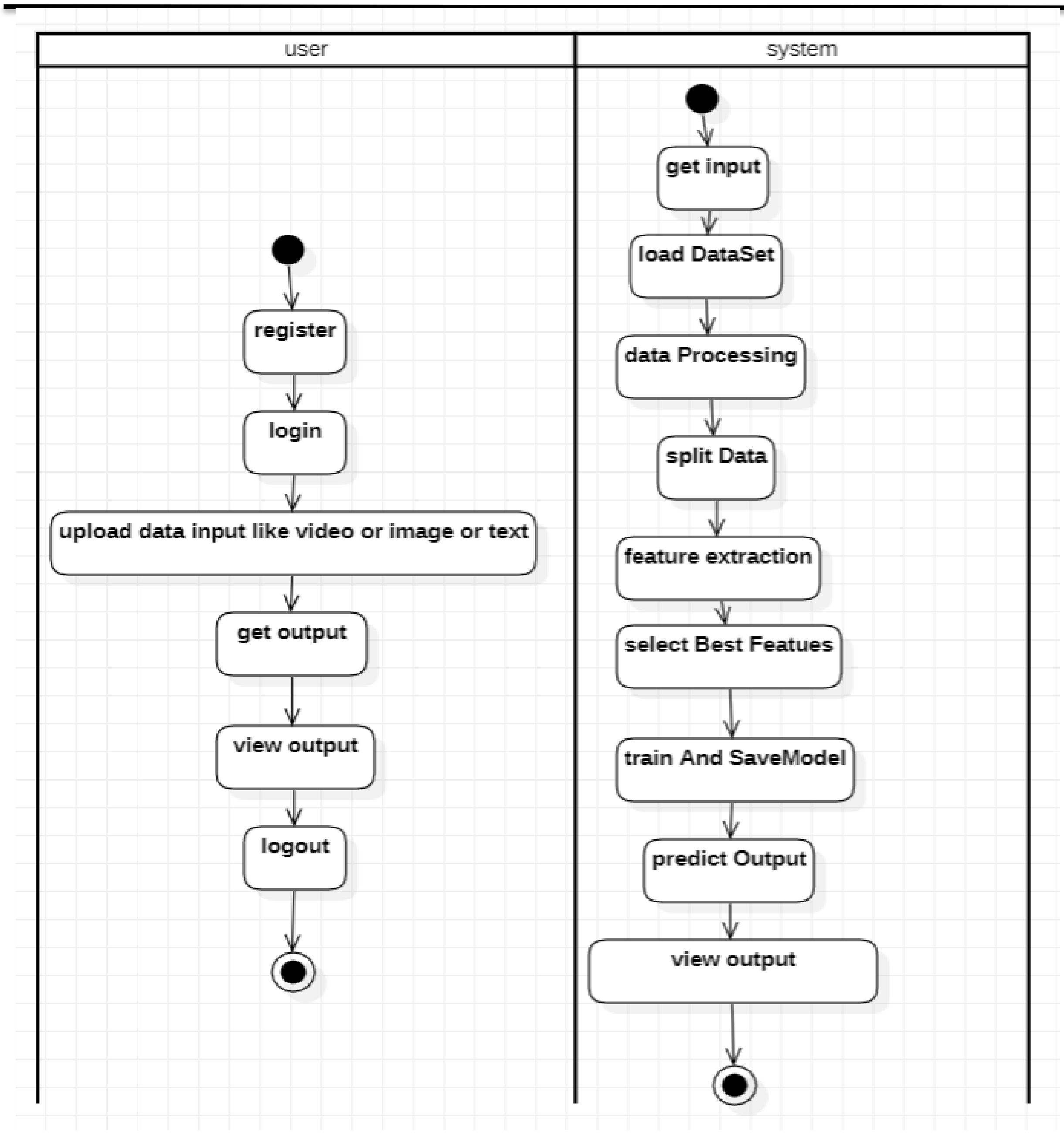


Figure 5. 5 Activity Diagram

5.4.5 DATA FLOW DIAGRAM

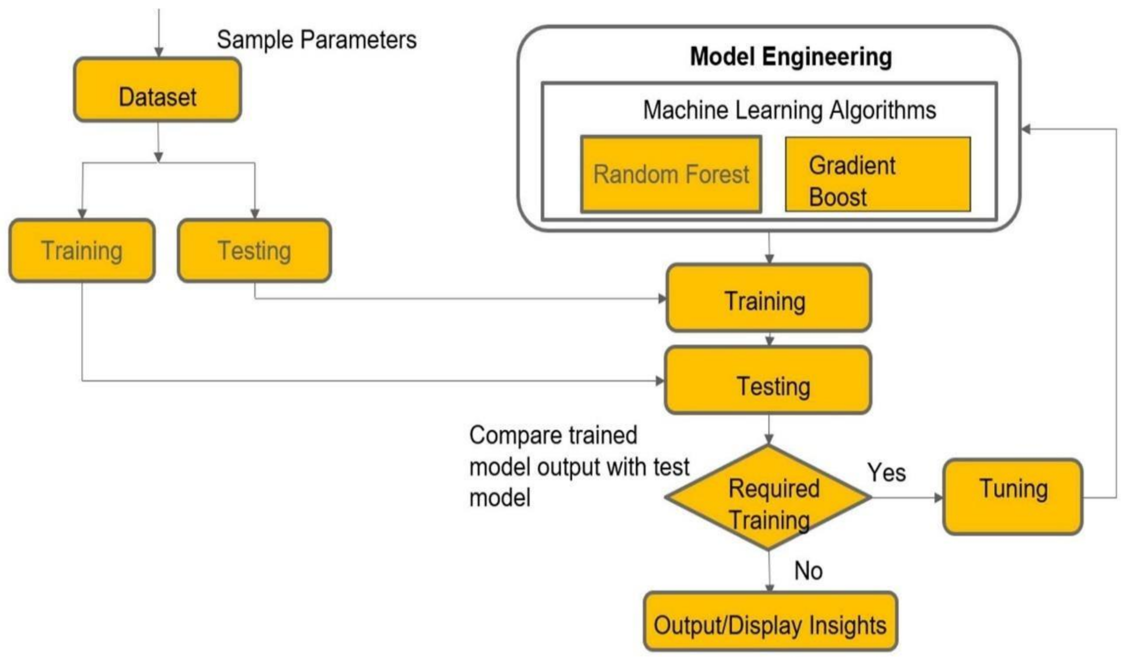


FIG 5.4.4 DATA FLOW DIAGRAM

CHAPTER-6 SYSTEM IMPLEMENTATION

To conduct studies and analyses of an operational and technological nature, and To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

6.1 INPUT AND OUTPUT DESIGNS

6.1.1 LOGICAL DESIGN

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

6.1.2 PHYSICAL DESIGN

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

6.2 INPUT & OUTPUT REPRESENTATION

6.2.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?

- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

6.2.2 OBJECTIVES

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

a. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that

b. people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
c. Select methods for presenting information.
d. Create document, report, or other formats that contain information produced by the system.

6.3 Code

# App.py

```
from flask import Flask, render_template, request
import pickle
import numpy as np
from database import *
from sklearn.preprocessing import LabelEncoder
import joblib
app = Flask(__name__,static_url_path='/static')

# Load the machine learning model

# Load saved model and scaler


@app.route('/p')
def p():
    return render_template('index.html')

@app.route('/')
def m():
    return render_template('main.html')

@app.route('/l')
def l():
    return render_template('login.html')

@app.route('/h')
def h():
    return render_template('home.html')

@app.route('/r')
def r():
    return render_template('register.html')

@app.route('/m')
def menu():
    return render_template('menu.html')


@app.route("/register",methods=['POST','GET'])
def signup():
    if request.method=='POST':
        username=request.form['username']
        email=request.form['email']
        password=request.form['password']
        status = user_reg(username,email,password)
        if status == 1:
            return render_template("/login.html")
        else:
            return
render_template("/register.html",m1="failed")


@app.route("/login",methods=['POST','GET'])
def login():
    if request.method=='POST':
        username=request.form['username']
        password=request.form['password']
        status = user_loginact(request.form['username'],
request.form['password'])
        print(status)
        if status == 1:
            return render_template("/home.html", m1="sucess")
        else:
            return render_template("/login.html", m1="Login Failed")

@app.route('/predict', methods=['POST'])
def predict():
    sr = float(request.form['sr'])  # Snoring Rate
    rr = float(request.form['rr'])  # Respiration Rate
    t = float(request.form['t'])  # Body Temperature
    lm = float(request.form['lm'])  # Limb Movement
    bo = float(request.form['bo'])  # Blood Oxygen
    rem = float(request.form['rem'])  # Eye Movement
    sr2 = int(request.form['sr.1'])  # Sleeping Hours (renamed to
avoid duplicate 'sr')
    hr = int(request.form['hr'])  # Heart Rate
    # Load saved model and scaler
    model = joblib.load("best_model_stress.pkl")

    #
    input_data = np.array([[sr, rr, t, lm, bo, rem, sr2, hr]])
    prediction_result = model.predict(input_data)[0]

    print(f"Predicted stress level: {prediction_result}")
            # Convert input to DataFrame


    if prediction_result == 0:
        op1 = "This Person is not Stressed!"
    if prediction_result == 1:
        op1 = "This Person is  Low Stress!"
    if prediction_result == 2:
        op1 = "This Person is  Medium Stressed!"
    if prediction_result == 3:
        op1 = "This Person is  Highly Stressed!"
    return render_template("result.html", op1=op1)

if __name___== "__main__":
    app.run(debug=True, port=5112)
```

# algocode.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import joblib

# Load the dataset
file_path = "Stress Data_C.csv"
data = pd.read_csv(file_path)

# Split data into features and target
X = data.drop(columns=["stress_level"])
y = data["stress_level"]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Define models
models = {
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "SVM": SVC()
}

# Train models and evaluate accuracy
accuracy_results = {}
best_model = None
best_accuracy = 0

for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_results[model_name] = accuracy

    # Save the best model
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model = model

# Save the best model
best_model_path = "best_model_stress.pkl"
joblib.dump("best_model_stress.pkl")

# Plot accuracy comparison
plt.figure(figsize=(10, 6))
plt.bar(accuracy_results.keys(), accuracy_results.values(),
color=['blue', 'green', 'red', 'purple'])
plt.xlabel("Machine Learning Models")
plt.ylabel("Accuracy")
plt.title("Accuracy Comparison of Models")
plt.ylim(0.8, 1.0)
plt.show()

# Load best model and predict
best_model_loaded = joblib.load(best_model_path)
input_data = np.array([[93.8, 91.84, 89.84, 25.68, 1.84, 74.2, 1,
10]])
prediction_result = best_model_loaded.predict(input_data)[0]

print(f"Predicted stress level: {prediction_result}")
```

# database.py

```
import sqlite3

import hashlib

import datetime

import MySQLdb

from flask import session

from datetime import datetime

import matplotlib.pyplot as plt

import numpy as np

import argparse

import cv2
import os

import numpy as np

 import os

 import cv2

import pandas as pd

 def db_connect():

    _conn = MySQLdb.connect(host="localhost", user="root",

                          passwd="root", db="heart")

    c = _conn.cursor()

          return c, _conn


# ----------------------------------register-----------------------------------------------------------------------

def user_reg(username, email,password):

    try:

        status=user_loginact(username, password)

        if status==1:

            return 0

        c, conn = db_connect()

        print(username, password, email)

        j = c.execute("insert into user (username,email,password) values ('"+username +

                    "','"+email+"','"+password+"')")

        conn.commit()

        conn.close()

        print(j)

        return j

    except Exception as e:

        print(e)
        return(str(e))
# -----------------------------------------Login -----------------------------------------
def user_loginact(username, password):
    try:
        c, conn = db_connect()
        j = c.execute("select * from user where username='" +
                    username+"' and password='"+password+"'")
        data = c.fetchall()
        print(data)


        c.fetchall()
        conn.close()
        return j
    except Exception as e:
        return(str(e))
if _name___== "_main_":
    print(db_connect())
```

CHAPTER -7
IMPLEMENTATION

INTRODUCTION

The implementation of Stress detection in IT Employees using ML gives a clear idea to know the level of stress that an IT Employee falls under the category like positive, Low, Medium, High and Extreme kinds of Stress. By identifying the stress in the initial stages can prevent the causes of stress. It provides a way to know their levels of stress and it generates a good practices and advices to get rid of those kinds of stress and serve the purpose of live. It depicts a valid information that need to be considered to relieve from the stress.

8.1 IMPLEMENTATION OF KEY FUNCTIONS

To implement a model to identify the kinds of stress that a person holds, it comprises of many attributes like Snoring Rate, Body Temperature, Blood Oxygen, Respiration Rate, Sleeping Hours, Heart Rate, Headache, Working Hours. For each attribute it has certain levels if falls under that branch of a tree it gives that kind of a fruit which is similar to the type of stress. This is how the data falls under the range of following:

The Categories of stress follows as:

0 – Positive Stress

1 – Low Stress

2 – Medium Stress

3 – High Stress

4 – Extreme Stress

0	1	2	3	4
47-50	50-60	61 -77	78-94	95-99

Fig 7.1: Snoring Rate Range

0	1	2	3	4
97-100	94-96	92-93	91-90	85-89

Fig 7.2: Body Temperature Range

0	1	2	3	4
95-97	94-95	90-91	88-89.6	82-87

Fig 7.3: Blood Oxygen Range

0	1	2	3	4
16-18	19-20	20-22	22-25	26-29

Fig 7.4: Respiration Rate Range

0	1	2	3	4
7-10	5-6	2.8-4.7	0.7-1.8	0

Fig 7.5: Sleeping Hours Range

0	1	2	3	4
50-55	56-59	60-64	68-74	75-84

Fig 7.6: Heart Rate Range

0	1	2	3	4
0	0	0	1	1

Fig 7.7: Headache Range

0	1	2	3	4
4-5	6-7	8-9	10-12	13-15

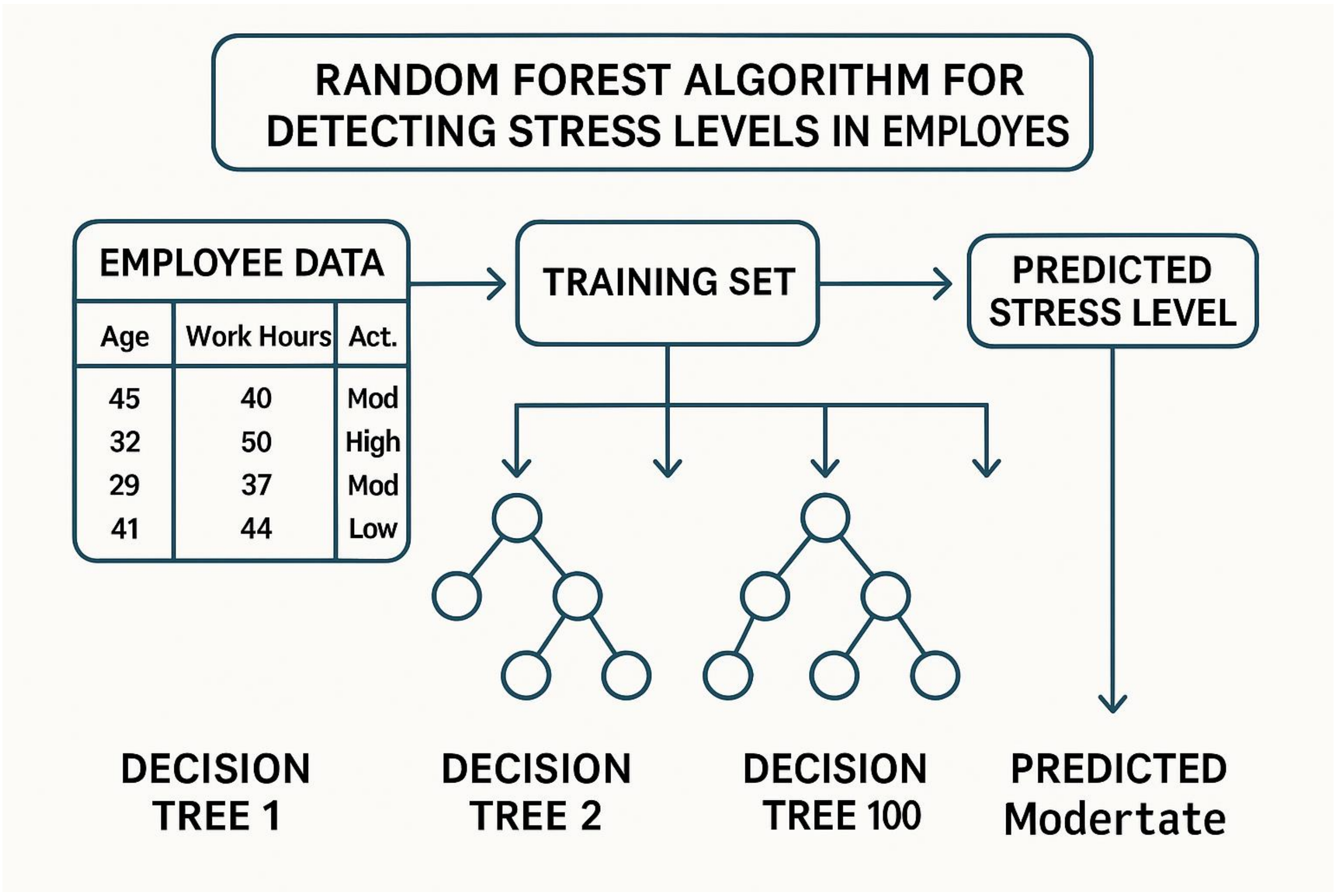
Fig 7.8: Working Hours Range

The various algorithms used to measure the level of stress are Random Forest, Gradient Boost, Support Vector Machine, Decision Tree, Naïve Bayes, K-Nearest Neighbours. Among these Random Forest provides higher rate of accuracy as compared to the remaining algorithms.

Random Forest Algorithm:

It is a supervised machine learning algorithm. It is made up of many decision trees at every instance it takes a route to generate a valid output. The Random Forest algorithm can be used for each classification resembles the frequency of repetition as well as regression problems results with the mean of the data.

The larger number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. It is also preferable to take small amount of data for training as compared to other algorithms. Even if there is a loss of data in large amount still it gives high accuracy by running efficiently. As the target variable is mentioned in the dataset and it gets mould itself to adopt changes by the model and predicts the output as per the behaviour of the data.



User Interface

1. In this project render is used to make the HTML, CSS and JavaScript files to create a use interaction, based environment by using the unique link anyone can be detected their levels of stress. The render environment provides web services to deploy their model and designing the input platform.

2. It is a user-friendly environment to make dynamic changes in the render. The form validation is incorporated within the html file to avoid unnecessary inputs.

CHAPTER-8
SYSTEM TESTING

8.1 INTRODUCTION:

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1 A successful test is one that uncovers an as yet undiscovered error.
2 A good test case is one that has probability of finding an error, if it exists.
3 The test is inadequate to detect possibly present errors.
4 The software more or less confirms to the quality and reliable standards.

8.2 LEVELS OF TESTING

Code testing:

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing:

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing:

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1 Black Box Testing
2 White Box Testing

8.2.1 BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Input
Black Box
Output

The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.

- Regression testing – Regression testing is done after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

8.2.2 WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

WHAT DO YOU VERIFY IN WHITE BOX TESTING?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

HOW DO YOU PERFORM WHITE BOX TESTING?

To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

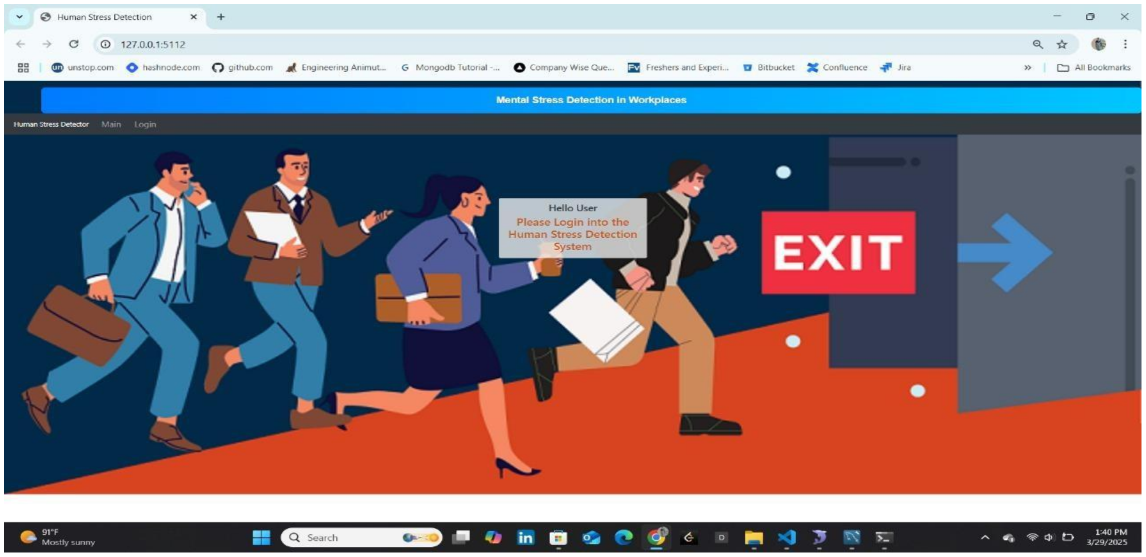
The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

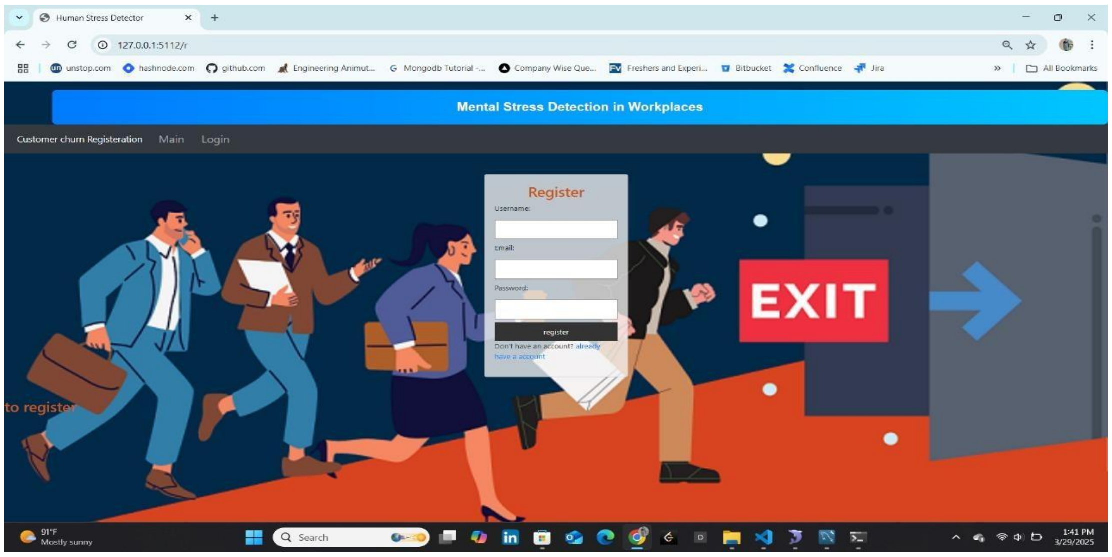
The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

CHAPTER-10
RESULTS AND SCREENSHOTS

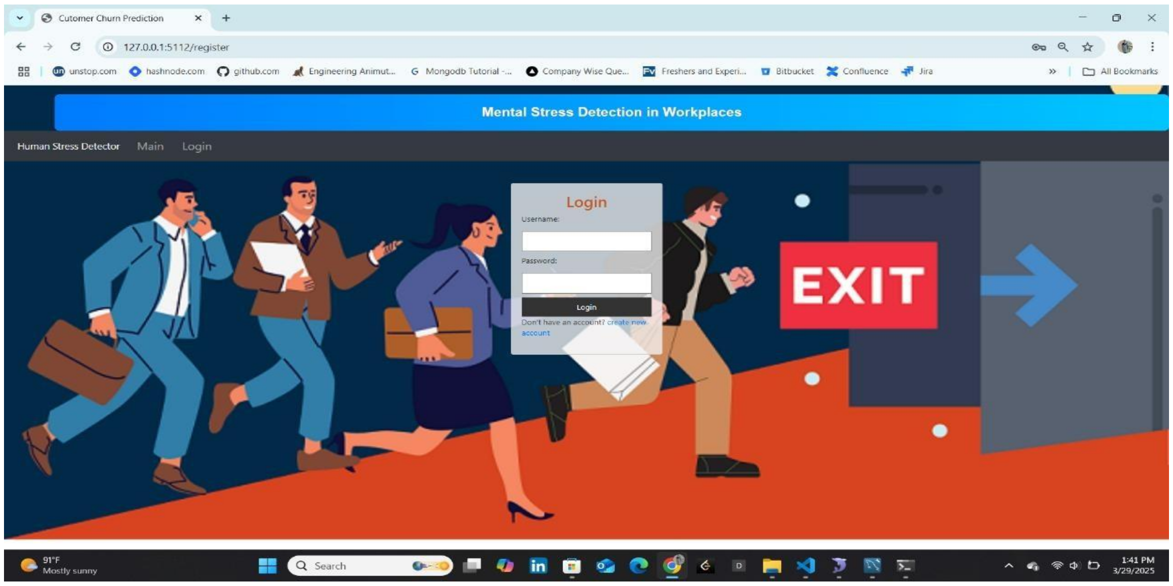
Home Page:


Mental Stress Detection in Workplaces
Human Stress Detector Main Login
Hello User
Please Login into the Human Stress Detection System
EXIT

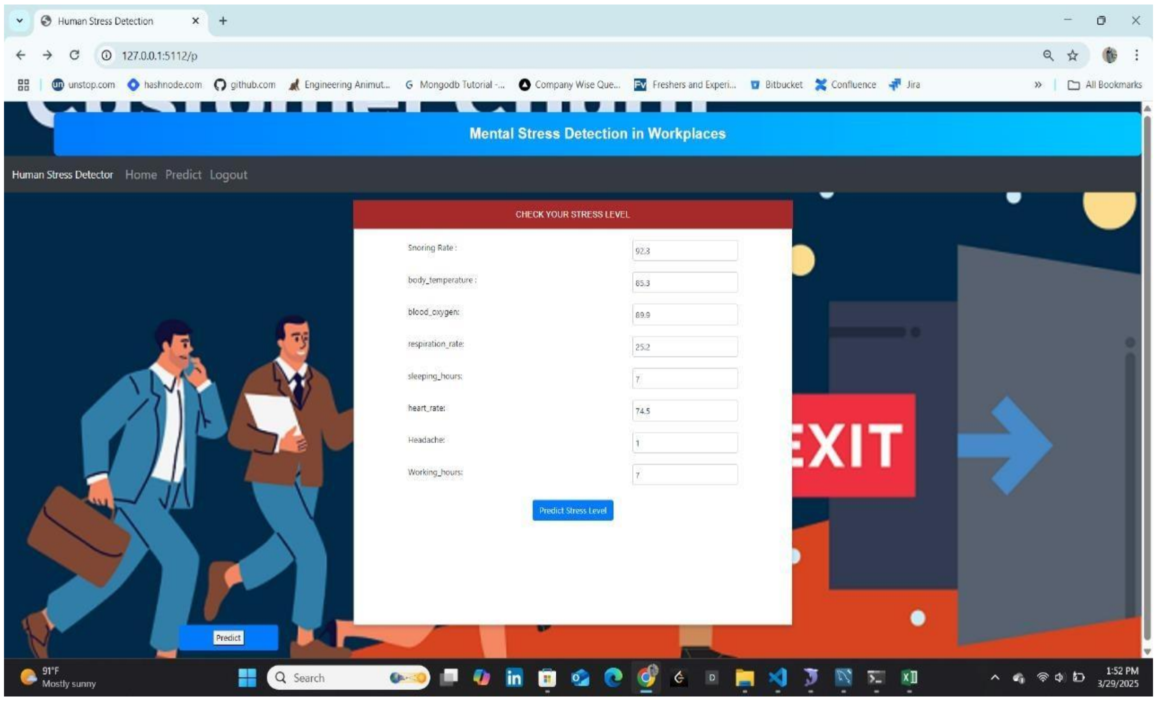
Register Page:


Mental Stress Detection in Workplaces
Customer churn Registeration Main Login
Register
Username:
Email:
Password:
register
Don't have an account? already have a account
to register
EXIT

Login Page:

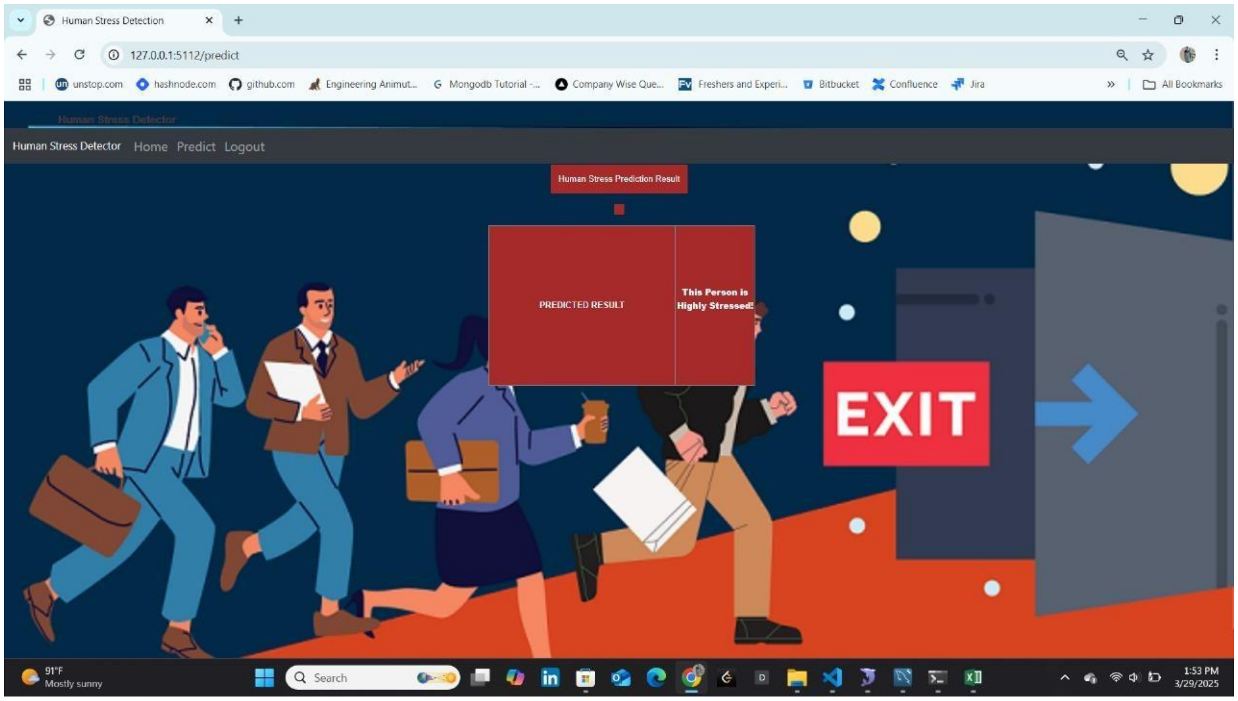


Stress detector page:



Snoring Rate :	92.3
body_temperature :	85.3
blood_oxygen:	89.9
respiration_rate:	25.2
sleeping_hours:	7
heart_rate:	74.5
Headache:	1
Working_hours:	7

Result page:


Human Stress Detection
127.0.0.1:5112/predict
Human Stress Detector
Home
Predict
Logout
Human Stress Prediction Result
PREDICTED RESULT
This Person is Highly Stressed!
EXIT

CHAPTER-9 CONCLUSION

GENERAL

Stress is a significant issue faced by IT professionals, with a high percentage of individuals experiencing stress symptoms. Machine learning models have been developed to detect stress levels in IT employees using various methods such as surveys, physiological recordings, and digital footprints. The stress detection model has been trained and tested using heterogeneous machine learning algorithms, and it has been deployed using the Flask framework.The model represents stress levels on a scale ranging from 0 (rest) to 4 (extreme stress), and it can identify different types of stress, including acute, episodic, chronic, and toxic stress. The implementation of the stress detection model involves dataset selection, data preprocessing, algorithm selection, and prediction of stress levels. The model considers various parameters such as snoring rate, body temperature, blood oxygen, and respiration rate to determine stress levels.

SUMMARY OF WORK

1. The project aims to detect stress levels in IT employees using machine learning algorithms and provide suggestions to manage stress effectively.

2. Stress is a major issue for IT professionals due to factors such as increased competition, high demand, prolonged working hours, and the evolution of technologies.

3. The model is trained and tested with heterogeneous machine learning algorithms to efficiently detect stress levels in IT employees.

4. The stress levels are represented in terms of 0-Rest, 1-Low Stress, 2-Medium Stress, and 3-High Stress, allowing IT employees to identify their stress levels

CHAPTER-10

Future Enhancements:

- **Integration with Smart Wearables:** Automating data collection from **smartwatches, fitness bands, or IoT-based devices**.
- **Enhanced AI Models:** Implementing **CNNs, LSTMs, or hybrid models** for even more accurate predictions.
- **Real-time Monitoring & Alerts:** Providing **instant stress alerts** and recommendations based on continuous monitoring.
- **Personalized Stress Management Plans:** Offering tailored **relaxation techniques, meditation guides, and lifestyle recommendations** based on stress levels.

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.
2) Nisha Raichur, Nidhi Lonakadi, Priyanka Mural, “Detection of Stress Using Image Processing and Machine Learning Techniques”, vol.9, no. 3S, July 2017.
3) 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.
4) T. Jick and R. Payne, “Stress at work,” Journal of Management Education, vol. 5, no. 3, pp. 50-56, 1980.
5) Bhattacharyya, R., & Basu, S. (2018). Retrieved from ‘The Economic Times’.
6) OSMI Mental Health in Tech Survey Dataset, 2017.
7) Communications,N..World health report. 2001.URL: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.
9) 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.
10) 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.

11) 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.

12) Liu, D., Ulrich, M.. Listen to your heart: Stress prediction using consumer heart rate sensors 2015;.

13) 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.

14) 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.