**Multi Class Stress Detection Through Heart Variability A Deep Neural Network Based Study***A major project report submitted in partial fulfillment of the requirements* *for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

*by*

**CH.AKHILA (20S41A0523)**

**A.SAMYUKTHA (20S41A0506)**

**M.INDHU (20S41A0560)**

**A.ROHITH (20S41A0510)**

**Under the Guidance of**

**MRS.K.ANITHA**

**Associate Professor**

**Department of Computer Science & Engineering**

**VAAGESWARI COLLEGE OF ENGINEERING**

**Accredited by NAAC with A+ Grade**

**(Affiliated to JNTU Hyderabad &Approved by AICTE New Delhi)**

**Ramakrishna colony, Karimnagar-505527**

**December 2024**

**Department of Computer Science & Engineering**

**VAAGESWARI COLLEGE OF ENGINEERING**

**Accredited by NAAC with A+ Grade**

**(Affiliated to JNTU Hyderabad &Approved by AICTE New Delhi)**

 **Ramakrishna colony, Karimnagar-505527**

**CERTIFICATE**

# This is to certify that the project report entitled “Multi Class Stress Detection Through Heart Variability A Deep Neural Network Based Study”.submitted by the following students in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in CSE and is a bonafide record of the work performed by.

**CH.AKHILA (20S41A0523)**

**A.SAMYUKTHA (20S41A0506)**

**M.INDHU (20S41A0560)**

**A.ROHITH (20S41A0510)**

**Internal Guide Head Of The Department**

**MRS.ANITHA Dr. N. CHANDRAMOULI**

**Associate Professor Associate Professor**

## Principal External Examiner Dr.CH.SRINIVAS

**DECLARATION**

We hereby declare that the project titled “**Multi Class Stress Detection Through Heart Variability A Deep Neural Network Based Study”** submitted to Vaageswari College of Engineering, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the Degree of Bachelor of Technology in CSE is a result of original research carried-out in this work. It is further declared that the report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

1. **Name of the Student :AKHILA CHOLLETI**

**Hall ticket Number : 20S41A0523**

**2. Name of the Student : SAMYUKTHA ANTAGIRI**

**Hall ticket Number : 20S41A0506**

**3.Name of the Student : INDHU MADADI**

**Hall ticket Number : 20S41A0560**

**4.Name of the Student : ROHITH ARUMULLA**

**Hall ticket Number : 20S41A0510**

Date:

**ACKNOWLEDGEMENT**

We wish to pay our sincere thanks to **Dr. Ch. Srinivas**, Principal, Vaageswari College of Engineering, Karimnagar, for providing all required facilities and his support during the Seminar.

We would like to thank **Dr. N. Chandramouli,** Associate. Professor and HOD of the Computer Science and Engineering Department for his valuable suggestions during the project.

We sincerely extend our thanks to the project guide, **Mrs.K.Anitha**, Associate Professor of Computer Science and Engineering Department, for sparing his valuable time in guiding the project work and giving feedback with many useful suggestions during the project work.

We are also conveying our heartfelt thanks to the Institute authority, Department, Library, and Laboratory staff of Vaageswari College of Engineering for their co-operation during our seminar. We thank our beloved friends for their help and encouragement regarding the concepts and presentation

**CH.AKHILA (20S41A0523)**

**A.SAMYUKTHA (20S41A0506)**

**M.INDHU (20S41A0560)**

**A.ROHITH (20S41A0510)**

**INDEX**

**TOPICS Page No’s**

* **Certificates**
* **Acknowledgement**
* **Abstract**
* **Figures/Tables**

[**CHAPTER-1: INTRODUCTION**](#_TOC_250005) **3**

[**CHAPTER-2: LITERATURE SURVEY**](#_TOC_250004) **6**

[**CHAPTER-3: SYSTEM ANALYSIS**](#_TOC_250003)

* 1. **Existing System 8**
  2. **Proposed System 8**

[**CHAPTER-4: SYSTEM REQUIREMENTS**](#_TOC_250002)

* 1. **Hardware Requirements 10**
  2. **Software Requirements 10**

[**CHAPTER-5: SYSTEM STUDY**](#_TOC_250001)

* 1. **Feasibility Study 11**
  2. **Feasibility Analysis 11**

[**CHAPTER-6: SYSTEM ARCHITECTURE**](#_TOC_250000)

* 1. **ARCHITECTURE 12**
  2. **UML Diagrams 13-18**

**CHAPTER-7: INPUT AND OUTPUT DESIGN**

* 1. **Input Design 19**
  2. **Output Design**

**CHAPTER-8: IMPLEMENTATION**

* 1. **MODULES 21**
     1. **Module Description 21**

**CHAPTER-9: SOFTWARE ENVIRONMENT**

* 1. **PYTHON 22-45**
  2. **Source Code 46-52**

**CHAPTER-10: RESULTS/DISCUSSIONS**

* 1. **System Test 53-57**
  2. **Output Screens 58-59**

**CHAPTER-11: CONCLUSION 60**

**CHAPTER-12: REFERENCES/BIBLIOGRAPHY 61-62**

# LIST OF FIGURES

**FIGURES/TABLES**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Figures/Tables** | **Page No.** |
| 1 | ARCHITECTURE | 12 |
| 2 | UML Diagrams | 14-18 |
| 3 | Compiling and interpreting python source code | 46-53 |
| 4 | Output Screens | 58-59 |

**Multi Class Stress Detection Through Heart Rate**

**Variability A Deep Neural Network Based Study**

**ABSTRACT**

Stress is a natural human reaction to demands or pressure, usually when perceived as harmful or/and toxic. When stress becomes constantly overwhelmed and prolonged, it increases the risk of mental health and physiological uneasiness. Furthermore, chronic stress raises the likelihood of mental health plagues such as anxiety, depression, and sleep disorder. Although measuring stress using physiological parameters such as heart rate variability (HRV) is a common approach, how to achieve ultra-high accuracy based on HRV measurements remains as a challenging task. HRV is not equivalent to heart rate. While heart rate is the average value of heartbeats per minute, HRV represents the variation of the time interval between successive heartbeats. The HRV measurements are related to the variance of RR intervals which stand for the time between successive R peaks. In this study, we investigate the role of HRV features as stress detection bio-markers and develop a machine learning-based model for multi-class stress detection. More specifically,

a convolution neural network (CNN) based model is developed to detect multi-class stress, namely, *no stress, interruption stress, and time pressure stress*, based on both time- and frequency-domain features of HRV.Validated through a publicly available dataset, SWELL−KW, the achieved accuracy score of our model hasreached 99.9% (*Precision*=*1*, *Recall*=*1*, *F1*−*score*=*1*, and *MCC*=*0.99*), thus outperforming the existingmethods in the literature. In addition, this study demonstrates the effectiveness of essential HRV features forstress detection using a feature extraction technique, i.e., analysis of variance.

**CHAPTER 1**

**INTRODUCTION**

Physical or mental imbalances caused by noxious stimuli trigger stress to maintain homeostasis. Under chronic stress, the sympathetic nervous system becomes overactive, leading to physical, psychological, and behavioral abnormalities [1]. Stress levels are often measured using subjective methods to extract perceptions of stress. Stress level measurement based on collected heart rate viability (HRV) data can help to remove the presence of stress by observing its effects on the autonomic nervous system (ANS) [2].

Typically, people with anxiety disorders have chronically lower resting HRV compared with healthy people. As revealed in [2] and [3], HRV increases with relaxation and decreases with stress. Indeed, HRV is usually higher when a heart is beating slowly and vice versa. Therefore, heart rate and HRV generally have an inverse relationship [2], [3]. HRV varies over time based on activity levels and the amount of work-related stress.

Furthermore, stress is usually associated with a negative notion of a person and is considered to be a subjective feeling of human beings that might affect emotional and physical well-being. It is described as a psychological and biological reaction to internal or external stressors [4], including a biological or chemical agent and environmental stimulation that induce stress in an organism [5]. On a molecular scale, stress impacts the ANS [6], which uses sympathetic and parasympathetic components to regulate the cardiovascular system. The sympathetic component in a human body [7] works analogously to a car’s gas pedal. It activates the fight-or-flight response, giving the body a boost of energy to respond to negative influences. In contrast, the parasympathetic component is the brake for a body. It stimulates the body’s *rest and digests* reaction by relaxing the body when a threat has passed. Given the fact that the ANS regulates the mental stress level of a human being, physiological measurements such as electrocardiogram (ECG), electromyogram (EMG), galvanic skin response (GSR), HRV, heart rate, blood pressure, breath frequency, and respiration rate can be used to assess mental stress [8].

ECG signals are commonly adopted to extract HRV [9]. HRV is defined as the variation across intervals between consecutive regular RR intervals,1 and it is measured by determining the length between two successive heartbeat peaks from an ECG reading. Conventionally, HRV has been accepted as a term to describe variations of both instantaneous heart rate and RR intervals [12].

Obtaining HRV from ECG readings requires clinical settings and specialized technical knowledge for data interpretation. Thanks to the recent technological advances on the Internet of medical things (IOMT) [17], it is possible to deploy a commercially available wearable or non-wearable IOMT devices to monitor and record heart rate measurements.

Based on ECG data analysis (or HRV features, various machine learning (ML) and deep learning (DL) algorithms have been developed in recent years for stress prediction [20], [21], [22], [23], [24], [25], [26], [27] (see more details in Sec. II). Among the publicly available datasets for stress detection, SWELL−KW developed in [13] and [14] one of the two most popular ones. However, none of the existing ML and DL studies based on the SWELL−KW dataset for multi-class stress classification have achieved ultra-high accuracy, especially for multi-class stress level classification [15], [16]. Therefore, there exists a research gap on developing novel ML models which are able to achieve ultra-high accurate prediction.

Motivated by various existing applied ML and DL based studies on HRV feature processing for stress level classifications, we have designed and developed a one-dimensional convolutional neural network (1D CNN) model for multi-class stress classification and demonstrate its superiority over the state-of-the-art models based on the SWELL-KW dataset in term of prediction accuracy. More specifically, we have performed studies on stress detection using both traditional machine learning algorithms and/or multi-layer perceptron (MLP) algorithms which are inspired from the fully connected neural network (FCNN) architecture. In our work, we have developed a 1D CNN model which is based on the convolution operation. CNN reduces number of training parameters as MLP takes vector as input and CNN takes tensor as input so that CNN can understand spatial relation.

While the accuracy achieved with full features is nearly 100%, we have also introduced a feature reduction algorithm based on *analysis of variance (ANOVA)* F-test and demonstrate that it is possible to achieve an accuracy score of 96.5% with less than half of the features that are available in the SWELL−KW dataset. Such a feature extraction reduces the computational load during the model training phase.

In a nutshell, the novelty and the main contributions of this study are summarized as follows:

• We have developed a novel 1D CNN model to detect multi-class stress status with outstanding performance, achieving 99.9% accuracy with a *Precision, F1-score*, and *Recall* score of 1.0 respectively and a *Matthews correlation coefficient (MCC)* score of 99.9%. We believe this is the first study that achieves such a high score of accuracy for multi-class stress classification.

• Furthermore, we reveal that not all 34 HRV features are necessary to accurately classify multi-class stress. We have performed feature optimization to select an optimized feature set to train a 1D CNN classifier, achieving a performance score that beats the existing classification models based on the SWELL-KW dataset.

• Our model with selected top-ranked HRV features does not require resource-intensive computation and it achieves also excellent accuracy without sacrificing critical information.

The remainder of the paper is organized as follows. After summarizing related work and pointing out the distinction between our work and the existing work in Sec. II, we introduce briefly the framework for stress status classification, dataset, and data preprocessing in Sec. III. Then the developed CNN model is presented in Sec. IV. Afterwards, Sec. V defines the performance metrics to evaluate the proposed classifier and Sec. VI presents the numerical results. Further discussions are provided in Sec. VII. Finally, the paper is concluded in Sec. VIII

**CHAPTER-2**

**LITERATURE SURVEY**

**Literature Survey**

**Title**:Multi Class Stress Detection Through Heart Rate Variability

**Authors:** JON ANDREAS MORTENSEN, MARTIN EFREMOV MOLLOV**.**

**Abstract:** Stress is a natural human reaction to demands or pressure, usually when perceived as harmful or/and toxic. When stress becomes constantly overwhelmed and prolonged, it increases the risk of mental health and physiological uneasiness. Furthermore, chronic stress raises the likelihood of mental health plagues such as anxiety, depression, and sleep disorder. Although measuring stress using physiological parameters such as heart rate variability (HRV) is a common approach, how to achieve ultra-high accuracy based on HRV measurements remains as a challenging task. HRV is not equivalent to heart rate. While heart rate is the average value of heartbeats per minute, HRV represents the variation of the time interval between successive heartbeats. The HRV measurements are related to the variance of RR intervals which stand for the time between successive R peaks.

**Title:** Multi-Class Stress Detection Through Heart Rate Variability: A Deep Neural Network Based Study

**Authors:** AYAN CHATTERJEE, DEBASISH GHOSE

**Abstract:** ECG signals are commonly adopted to extract HRV [9]. HRV is defined as the variation across intervals between consecutive regular RR intervals,1 and it is measured by determining the length between two successive heartbeat peaks from an ECG reading. Conventionally, HRV has been accepted as a term to describe variations of both instantaneous heart rate.

In a nutshell, the novelty and the main contributions of this study are summarized as follows: • We have developed a novel 1D CNN model to detect multi-class stress status with outstanding performance, achieving 99.9% accuracy with a Precision, F1-score, and Recall score of 1.0 respectively and a Matthews correlation coefficient (MCC) score of 99.9%. We believe this is the first study that achieves such a high score of accuracy for multi-class stress classification. • Furthermore, we reveal that not all 34 HRV features are necessary to accurately classify multi-class stress. We have performed feature optimization to select an optimized feature set to train a 1D CNN classifier, achieving a performance score that beats the existing classification models based on the SWELL-KW dataset. • Our model with selected top-ranked HRV features does not require resource-intensive computation and it achieves also excellent accuracy without sacrificing critical information

**Title:** Multi-Class Stress Detection Through Heart Rate Variability through a CNN model for stress status

**Authors:** FRANK Y. LI, JON ANDREAS MORTENSEN

**Abstract:**

A CNN model for stress status classification In this section, we present the developed deep learning model for stress status classification. As shown on the right-side hand of Fig. 1, the model consists of feature ranking, feature extraction, and tress level classification.

**A. FEATURE RANKING AND EXTRACTION:**

Firstly, we rank the essential features based on their relevance to the classification task. To do so, the ANOVA [31] Ftest is adopted to select the significant features from the SWELL−KW dataset for feature ranking and extraction. ANOVA is a popular tool to perform a parametric statistical hypothesis test that assesses whether the means of two or more data samples (typically three or more) are from the same distribution or not. An F-statistic or F-test is a statistical test method that adopts ANOVA to calculate the ratio between variance values, such as variance from two different samples, or explained and unexplained variance. Furthermore, ANOVA can be used when one variable is numeric, and the other one is categorical, such as when a numerical input data and a classification outcome variable are compared in a classification task. In this study, we first employ all features for stress classification and then drop the minor significant features based on the importance of features (i.e., feature ranking) before performing the classification task. In the latter case, the training time is shortened while keeping the accuracy of the model.

**B. A CNN DL MODEL FOR STRESS CLASSIFICATION:**

The designed DL model for stress level classification is developed based on the conventional, well-known CNN architectures [32]. CNN is a powerful tool for automatic feature extraction and learning from 1D data sequences. The HRV features of the CNN architecture that are used in our model are illustrated in Tab. 1. For our model design, we retain a reasonable number of neurons in each layer based on the common heuristics (e.g., validation loss, hidden units are a fraction of the input). The CNN kernels slide over the components of the 1D input pattern during convolution. More specifically, our 1D CNN model consists of an input layer, multiple hidden layers, a max-pooling layer, a flattening layer, and an output layer, as depicted in Fig. 5. The input layer is a 1D convolutional layer, and it consists of 64 filters, a kernel of size 2, and a relative light unit (ReLU) activation function. The ReLU activation.

# CHAPTER-3

# SYSTEM ANALYSIS

**EXISTING SYSTEM**

For HRV data quality, a detailed review on data received from ECG and IoMT devices such as Elite HRV, H7, Polar, and Motorola Droid can be found in [18]. 23 studies indicated minor errors when comparing the HRV values obtained from commercially available IoMT devices with ECG instrument based measurements. In practice, such a small-scale error in HRV measurements is reasonable, as getting HRVs using portable IoMT devices is more practical, cost-effective, and no laboratory/clinical equipment is required [18], [19].

On the other hand, there have been a lot of recent research efforts on ECG data analysis to classify stress through ML and DL algorithms [20], [21], [22], [23]. Existing algorithms have focused mainly on binary (stress versus nonstress) and multi-class stress classifications. For instance, the authors in [4] classified HRV data into stressed and normal physiological states. The authors compared different ML approaches for classifying stress, such as naive Bayes, knearest neighbour (KNN), support vector machine (SVM), MLP, random forest, and gradient boosting. The best recall score they achieved was 80%. A similar comparison study was performed in [27], where the authors showed that SVM with radial basis function (RBF) provided an accuracy score of 83.33% and 66.66% respectively, using the time-domain and frequency-domain features ofHRV. Moreover, dimension reduction techniques have been applied to select best temporal and frequency domain features in HRV [24]. Binary classification, i.e., stressed versus not stressed, was performed using CNN in [25] through which the authors achieved an accuracy score of 98.4%. Another study, StressClick [26], employed a random forest algorithm to classify stressed versus not stressed based on mouse-click events, i.e., the gaze-click pattern collected from the commercial computer webcam and mouse.

In [14], tasks for multi-class stress classification (e.g., no stress, interruption stress, and time pressure stress) were performed using SVM based on the SWELL−KW dataset. The highest accuracy they achieved was 90%. Furthermore, another publicly available dataset, WESAD, was used in [27] for multi-class (amusement versus baseline versus stress) and binary (stress versus non-stress) classifications. In their investigations, ML algorithms achieved accuracy scores up to 81.65% for three-class categorization.

The authors also checked the performance of deep learning algorithms, where they achieved an accuracy level of 84.32% for three-class stress classification. Furthermore, it is worth mentioning that novel deep learning techniques, such as genetic deep learning convolutional neural networks (GDCNNs) [38], [39], have appeared as a powerful tool for two-dimensional data classification tasks. To apply GDCNN to 1D data, however, comprehensive modifications or adaptations are required and such a topic is beyond the scope of this paper.

**Disadvantages**

* Adaptive moment estimation (ADAM) optimizer as it is computationally efficient and claims less memory.
* Distinctive features are not considered from the new test samples, and the class label is resolved using all classification parameters estimated in training.

**Proposed System**

• We have developed a novel 1D CNN model to detect multi-class stress status with outstanding performance, achieving 99.9% accuracy with a *Precision, F1-score*, and *Recall* score of 1.0 respectively and a *Matthews correlation* *coefficient (MCC)* score of 99.9%. We believe this is the first study that achieves such a high score of

accuracy for multi-class stress classification.

• Furthermore, we reveal that not all 34 HRV features are necessary to accurately classify multi-class stress. We have performed feature optimization to select an optimized feature set to train a 1D CNN classifier, achieving a performance score that beats the existing classification models based on the SWELL-KW dataset.

• Our model with selected top-ranked HRV features does not require resource-intensive computation and it achieves also excellent accuracy without sacrificing critical information.

**Advantages:**The designed DL-based multi-class classifier is trained, tested, and validated with significant features and annotations (e.g., *no stress, interruption condition*, and *time pressure*) labeled by medical professionals.

**CHAPTER-4**

**SYSTEM REQUIREMENTS**

➢ **H/W System Configuration:-**

➢ Processor - Pentium –IV

➢ RAM - 4 GB (min)

➢ Hard Disk - 20 GB

➢ Key Board - Standard Windows Keyboard

➢ Mouse - Two or Three Button Mouse

➢ Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* **Operating system :** Windows 7 Ultimate.
* **Coding Language :** Python.
* **Front-End :** Python.
* **Back-End :** Django-ORM
* **Designing :** Html, css, javascript.
* **Data Base :** MySQL (WAMP Server).

**CHAPTER-5**

**SYSTEM `STUDY**

**2. SYSTEM STUDY**

**2.1 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.

# CHAPTER-6

# SYSTEM ARCHITECTURE

Service Provider

Login,

Train & Test Stroke Datasets,

View Healthcare Datasets Trained and Tested Accuracy in Bar Chart,

View Healthcare Datasets Trained and Tested Accuracy Results,

View Stress Prediction Type View Stress Prediction Type Ratio,

Download Predicted Data Sets,

View Stress Prediction Type Ratio Results,

View All Remote Users.

Accepting all Information

**Web Server**

Datasets Results Storage

Accessing Data

Process all user queries

**WEB Database**

Remote User

Tweet Server

Tweet Server

Tweet Server

**Store and retrievals**

REGISTER AND LOGIN,

PREDICT STRESS PREDICTION TYPE,

VIEW YOUR PROFILE.

**4.1 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 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:**

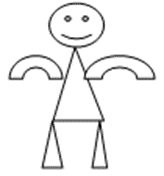
The Primary goals in the design of the UML are as follows:

* + 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 between those use cases can be depicted.

**service provider**

****

**remote user**

**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.



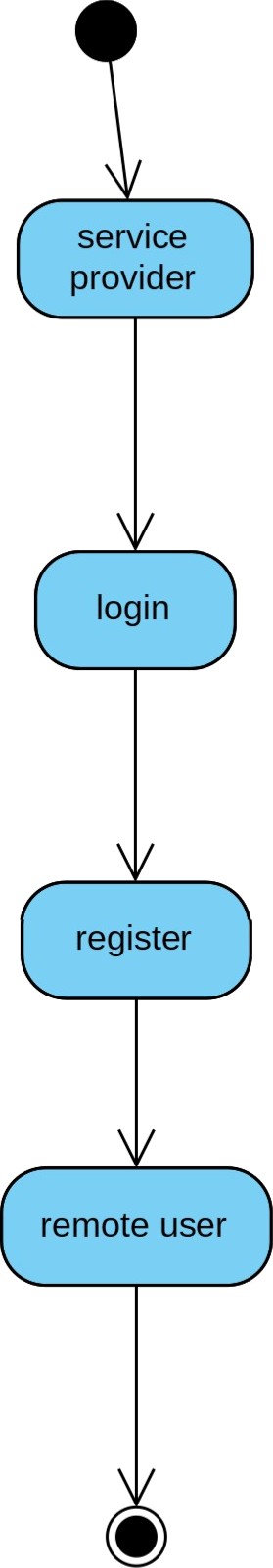
## SEQUENCE DIAGRAM:

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 used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. It is a type of behavioral diagram and we can depict both sequential processing and concurrent processing of activities using an activity diagram ie an activity diagram focuses on the condition of flow and the sequence in which it happens



**COMPONENT DIAGRAM**

A **component diagram** breaks down the actual system under development into various high levels of functionality. Each **component** is responsible for one clear aim within the entire system and only interacts with other essential elements

****

**DEPLOYMENT DIAGRAM**

A Deployment Diagram in software engineering is a type of Structural uml typw that shows the physical deployment of software components on hardware nodes. It illustrates the mapping of software components onto the physical resources of a system, such as servers, processors, storage devices, and network infrastructure.

****

**CHAPTER-7**

**INPUT AND OUTPUT DESIGN**

**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.

**OBJECTIVES**

1.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.

**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. 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 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.

* + 1. Select methods for presenting information.
    2. Create document, report, or other formats that contain information produced by the

system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

# CHAPTER-8

# IMPLEMENTATION

**Modules:**

**Service Provider**

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as Train & Test Stroke Datasets, View Healthcare Datasets Trained and Tested Accuracy in Bar Chart, View Healthcare Datasets Trained and Tested Accuracy Results, View Stress Prediction Type View Stress Prediction Type Ratio, Download Predicted Data Sets, View Stress Prediction Type Ratio Results, View All Remote Users.

**View and Authorize Users**

In this module, the admin can view the list of users who all registered. In this, the admin can view the user’s details such as, user name, email, address and admin authorizes the users.

**Remote User**

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN, PREDICT STRESS PREDICTION TYPE, VIEW YOUR PROFILE.

**CHAPTER-9**

**SOFTWARE ENVIRONMENT**

**1.1 PYTHON**

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting** **language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**1.2 History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

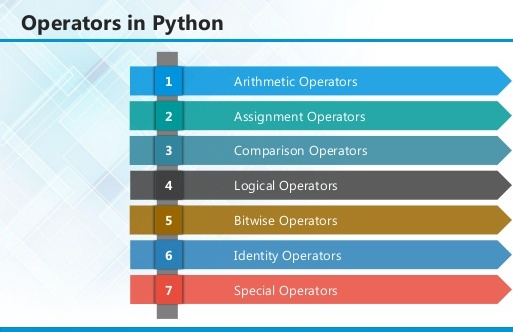
**1.3 Python Features**

Python's features include:

* **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
* **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases:** Python provides interfaces to all major commercial databases.
* **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.



**2.1 ARITHMETIC OPERATORS**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + Addition | Adds values on either side of the operator. | a + b = 30 |
| - Subtraction | Subtracts right hand operand from left hand operand. | a – b = -10 |
| \* Multiplication | Multiplies values on either side of the operator | a \* b = 200 |
| / Division | Divides left hand operand by right hand operand | b / a = 2 |
| % Modulus | Divides left hand operand by right hand operand and returns remainder | b % a = 0 |
| \*\* Exponent | Performs exponential (power) calculation on operators | a\*\*b =10 to the power 20 |
| // | Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity): | 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0 |

**2.2 ASSIGNMENT OPERATORS**

|  |  |  |
| --- | --- | --- |
|  | **Description** | **Example** |
| = | Assigns values from right side operands to left side operand | c = a + b assigns value of a + b into c |
| += Add AND | It adds right operand to the left operand and assign the result to left operand | c += a is equivalent to c = c + a |
| -= Subtract AND | It subtracts right operand from the left operand and assign the result to left operand | c -= a is equivalent to c = c - a |
| \*= Multiply AND | It multiplies right operand with the left operand and assign the result to left operand | c \*= a is equivalent to c = c \* a |
| /= Divide AND | It divides left operand with the right operand and assign the result to left operand | c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a |

|  |  |  |
| --- | --- | --- |
| %= Modulus AND | It takes modulus using two operands and assign the result to left operand | c %= a is equivalent to c = c % a |
| \*\*= Exponent AND | Performs exponential (power) calculation on operators and assign value to the left operand | c \*\*= a is equivalent to c = c \*\* a |
| //= Floor Division | It performs floor division on operators and assign value to the left operand | c //= a is equivalent to c = c // a |

**2.3 IDENTITY OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Evaluates to true if the variables on either side of the operator point to the same object and false otherwise | X is y here is results in 1 if id is |

**2.4 COMPARISON OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & Binary AND | Operator copies a bit to the result if it exists in both operands | (a & b) (means 0000 1100) |
| | Binary OR | It copies a bit if it exists in either operand. | (a | b) = 61 (means 0011 1101) |
| ^ Binary XOR | It copies the bit if it is set in one operand but not both. | (a ^ b) = 49 (means 0011 0001) |
| ~ Binary Ones Complement | It is unary and has the effect of 'flipping' bits. | (~a ) = -61 (means 1100 0011 in 2's complement form due to a signed binary number. |
| << Binary Left Shift | The left operands value is moved left by the number of bits specified by the right operand. | a << 2 = 240 (means 1111 0000) |
| >> Binary Right Shift | The left operands value is moved right by the number of bits specified by the right operand. | a >> 2 = 15 (means 0000 1111) |

**2.5 LOGICAL OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and Logical AND | If both the operands are true then condition becomes true. | (a and b) is true. |
| or Logical OR | If any of the two operands are non-zero then condition becomes true. | (a or b) is true. |
| not Logical NOT | Used to reverse the logical state of its operand. | Not(a and b) is false. |

**2.6** Membership Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Evaluates to true if it finds a variable in the specified sequence and false otherwise. | x in y, here in results in a 1 if x is a member of sequence y. |
| not in | Evaluates to true if it does not finds a variable in the specified sequence and false otherwise. | x not in y, here not in results in a 1 if x is not a member of sequence y. |

**Python Operators Precedence**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| \*\* | Exponentiation (raise to the power) |
| ~ + - | Complement, unary plus and minus (method names for the last two are +@ and -@) |
| \* / % // | Multiply, divide, modulo and floor division |
| + - | Addition and subtraction |
| >> << | Right and left bitwise shift |
| & | Bitwise 'AND' |  |
| ^ | | Bitwise exclusive `OR' and regular `OR' |  |
| <= < > >= | Comparison operators |  |
| <> == != | Equality operators |  |
| = %= /= //= -= += \*= \*\*= | Assignment operators |  |
| is is not | Identity operators |  |
| in not in | Membership operators |  |
| not or and | Logical operators |  |

**3.1 LIST**

The list is a most versatile data type 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"]

**Basic List Operations**

Lists respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| len([1, 2, 3]) | 3 | Length |
| [1, 2, 3] + [4, 5, 6] | [1, 2, 3, 4, 5, 6] | Concatenation |
| ['Hi!'] \* 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | Repetition |
| 3 in [1, 2, 3] | True | Membership |
| for x in [1, 2, 3]: print x, | 1 2 3 | Iteration |

**Built-in List Functions & Methods:**

Python includes the following list functions −

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [cmp(list1, list2)](https://www.tutorialspoint.com/python/list_cmp.htm)  Compares elements of both lists. |
| 2 | [len(list)](https://www.tutorialspoint.com/python/list_len.htm)  Gives the total length of the list. |
| 3 | [max(list)](https://www.tutorialspoint.com/python/list_max.htm)  Returns item from the list with max value. |
| 4 | [min(list)](https://www.tutorialspoint.com/python/list_min.htm)  Returns item from the list with min value. |
| 5 | [list(seq)](https://www.tutorialspoint.com/python/list_list.htm)  Converts a tuple into list. |

Python includes following list methods

|  |  |
| --- | --- |
| **SN** | **Methods with Description** |
| 1 | [list.append(obj)](https://www.tutorialspoint.com/python/list_append.htm)  Appends object obj to list |
| 2 | [list.count(obj)](https://www.tutorialspoint.com/python/list_count.htm)  Returns count of how many times obj occurs in list |
| 3 | [list. extend(seq)](https://www.tutorialspoint.com/python/list_extend.htm)  Appends the contents of seq to list |
| 4 | [list.index(obj)](https://www.tutorialspoint.com/python/list_index.htm)  Returns the lowest index in list that obj appears |
| 5 | [list.insert(index, obj)](https://www.tutorialspoint.com/python/list_insert.htm)  Inserts object obj into list at offset index |
| 6 | [list.pop(obj=list[-1])](https://www.tutorialspoint.com/python/list_pop.htm)  Removes and returns last object or obj from list |
| 7 | [list.remove(obj)](https://www.tutorialspoint.com/python/list_remove.htm)  Removes object obj from list |
| 8 | [list.reverse()](https://www.tutorialspoint.com/python/list_reverse.htm)  Reverses objects of list in place |
| 9 | [list.sort([func])](https://www.tutorialspoint.com/python/list_sort.htm)  Sorts objects of list, use compare function if given |

**TUPLE**

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 we 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 –

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 code is executed, it produces the following result −

tup1[0]: physics

tup2[1:5]: [2, 3, 4, 5]

**Updating Tuples:**

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

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

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:

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

print tup

del tup;

print "After deleting tup : "

print tup

**Basic Tuples Operations:**

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| len((1, 2, 3)) | 3 | Length |
| (1, 2, 3) + (4, 5, 6) | (1, 2, 3, 4, 5, 6) | Concatenation |
| ('Hi!',) \* 4 | ('Hi!', 'Hi!', 'Hi!', 'Hi!') | Repetition |
| 3 in (1, 2, 3) | True | Membership |
| for x in (1, 2, 3): print x, | 1 2 3 | Iteration |

**Built-in Tuple Functions**

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [**cmp(tuple1, tuple2)**](https://www.tutorialspoint.com/python/tuple_cmp.htm):Compares elements of both tuples. |
| 2 | [**len(tuple)**](https://www.tutorialspoint.com/python/tuple_len.htm):Gives the total length of the tuple. |
| 3 | [**max(tuple)**](https://www.tutorialspoint.com/python/tuple_max.htm):Returns item from the tuple with max value. |
| 4 | [**min(tuple)**](https://www.tutorialspoint.com/python/tuple_min.htm):Returns item from the tuple with min value. |
| 5 | [**tuple(seq)**](https://www.tutorialspoint.com/python/tuple_tuple.htm):Converts a list into tuple. |

**3.2 DICTIONARY**

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or 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 −

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

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

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

Result –

dict['Name']: Zara

dict['Age']: 7

**Updating Dictionary**

We 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 −

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']

Result −

dict['Age']: 8

dict['School']: DPS School

**Delete Dictionary Elements**

We 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 –

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

]

**Built-in Dictionary Functions & Methods –**

Python includes the following dictionary functions −

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [cmp(dict1, dict2)](https://www.tutorialspoint.com/python/dictionary_cmp.htm)  Compares elements of both dict. |
| 2 | [len(dict)](https://www.tutorialspoint.com/python/dictionary_len.htm)  Gives the total length of the dictionary. This would be equal to the number of items in the dictionary. |
| 3 | [str(dict)](https://www.tutorialspoint.com/python/dictionary_str.htm)  Produces a printable string representation of a dictionary |
| 4 | [type(variable)](https://www.tutorialspoint.com/python/dictionary_type.htm)  Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type. |

Python includes following dictionary methods −

**Calling a Function**

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print str

return;

# Now you can call printme function

printme("I'm first call to user defined function!")

printme("Again second call to the same function")

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

I'm first call to user defined function!

Again second call to the same function

**Function Arguments**

You can call a function by using the following types of formal arguments:

* Required arguments
* Keyword arguments
* Default arguments
* Variable-length arguments

**Scope of Variables**

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

Global variables Local variables

**Global vs. Local variables**

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope. Following is a simple example −

total = 0; # This is global variable.

# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total = arg1 + arg2; # Here total is local variable.

print "Inside the function local total : ", total

return total;

sum( 10, 20 );

print "Outside the function global total : ", total

**Result −**

Inside the function local total : 30

Outside the function global total : 0

**The *import* Statement**

The *import* has the following syntax:

import module1[, module2[,... moduleN]

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script −

**Packages in Python**

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

def Pots():

print "I'm Pots Phone"

Similar way, we have another two files having different functions with the same name as above −

* *Phone/Isdn.py* file having function Isdn()
* *Phone/G3.py* file having function G3()

Now, create one more file \_\_init\_\_.py in *Phone* directory −

* Phone/\_\_init\_\_.py

To make all of your functions available when you've imported Phone,to put explicit import statements in \_\_init\_\_.py as follows −

from Pots import Pots

from Isdn import Isdn

from G3 import G3

**RESULT:**

I'm Pots Phone

I'm 3G Phone

I'm ISDN Phone

**Printing to the Screen**

The simplest way to produce output is using the *print* statement where you can pass zero or more expressions separated by commas. This function converts the expressions you pass into a string and writes the result to standard output as follows −

print "Python is really a great language,", "isn't it?"

Result:

Python is really a great language, isn't it?

**Reading Keyboard Input**

Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard. These functions are −

* raw\_input
* input

The *raw\_input* Function

The *raw\_input([prompt])* function reads one line from standard input and returns it as a string (removing the trailing newline).

str = raw\_input("Enter your input: ");

print "Received input is : ", str

The *input* Function

The *input([prompt])* function is equivalent to raw\_input, except that it assumes the input is a valid Python expression and returns the evaluated result to you.

str = input("Enter your input: ");

print "Received input is : ", str

This would produce the following result against the entered input −

Enter your input: [x\*5 for x in range(2,10,2)]

Recieved input is : [10, 20, 30, 40]

Syntax

file object = open(file\_name [, access\_mode][, buffering])

Here are parameter details:

* **file\_name:** The file\_name argument is a string value that contains the name of the file that you want to access.
* **access\_mode:** The access\_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).
* **buffering:** If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

The *file* Object Attributes

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object:

Example

# Open a file

fo = open("foo.txt", "wb")

print "Name of the file: ", fo.name

print "Closed or not : ", fo.closed

print "Opening mode : ", fo.mode

print "Softspace flag : ", fo.softspace

This produces the following result −

Name of the file: foo.txt

Closed or not : False

Opening mode : wb

Softspace flag : 0

Reading and Writing Files

The *file* object provides a set of access methods to make our lives easier. We would see how to use *read()* and *write()* methods to read and write files.

The *write()* Method

The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.The write() method does not add a newline character ('\n') to the end of the string **Syntax**

fileObject.write(string);

Here, passed parameter is the content to be written into the opened file. **Example**

# Open a file

fo = open("foo.txt", "wb")

fo.write( "Python is a great language.\nYeah its great!!\n");

# Close opend file

fo.close()

The above method would create *foo.txt* file and would write given content in that file and finally it would close that file. If you would open this file, it would have following content.

Python is a great language.

Yeah its great!!

List of Standard Exceptions −

**What is Exception?**

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

**Handling an exception**

If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

* GadFly
* mSQL
* MySQL
* PostgreSQL
* Microsoft SQL Server 2000
* Informix
* Interbase
* Oracle
* Sybase
* The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:
* Importing the API module.
* Acquiring a connection with the database.
* Issuing SQL statements and stored procedures.
* Closing the connection

**9.2 SOURCE CODE:**

**Remote user.py**

from django.contrib import admin

from django.apps import AppConfig

class ClientSiteConfig(AppConfig):

name = 'Remote\_User'from django import forms

from Remote\_User.models import ClientRegister\_Model

class ClientRegister\_Form(forms.ModelForm):

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

email = forms.EmailField(required=True)

class Meta:

model = ClientRegister\_Model

fields = ("username","email","password","phoneno","country","state","city")

from django.db import models

# Create your models here.

from django.db.models import CASCADE

class ClientRegister\_Model(models.Model):

username = models.CharField(max\_length=30)

email = models.EmailField(max\_length=30)

password = models.CharField(max\_length=10)

phoneno = models.CharField(max\_length=10)

country = models.CharField(max\_length=30)

state = models.CharField(max\_length=30)

city = models.CharField(max\_length=30)

FID= models.CharField(max\_length=300)

MEAN\_RR= models.CharField(max\_length=300)

MEDIAN\_RR= models.CharField(max\_length=300)

SDRR= models.CharField(max\_length=300)

RMSSD= models.CharField(max\_length=300)

SDSD= models.CharField(max\_length=300)

SDRR\_RMSSD= models.CharField(max\_length=300)

HR= models.CharField(max\_length=300)

VLF= models.CharField(max\_length=300)

VLF\_PCT= models.CharField(max\_length=300)

LF= models.CharField(max\_length=300)

LF\_PCT= models.CharField(max\_length=300)

LF\_NU= models.CharField(max\_length=300)

HF= models.CharField(max\_length=300)

HF\_PCT= models.CharField(max\_length=300)

HF\_NU= models.CharField(max\_length=300)

TP= models.CharField(max\_length=300)

LF\_HF= models.CharField(max\_length=300)

HF\_LF= models.CharField(max\_length=300)

sampen= models.CharField(max\_length=300)

higuci= models.CharField(max\_length=300)

Prediction= models.CharField(max\_length=300)

names = models.CharField(max\_length=300)

SDRR\_RMSSD= request.POST.get('SDRR\_RMSSD')

HR= request.POST.get('HR')

VLF= request.POST.get('VLF')

VLF\_PCT= request.POST.get('VLF\_PCT')

LF= request.POST.get('LF')

LF\_PCT= request.POST.get('LF\_PCT')

LF\_NU= request.POST.get('LF\_NU')

HF= request.POST.get('HF')

HF\_PCT= request.POST.get('HF\_PCT')

HF\_NU= request.POST.get('HF\_NU')

TP= request.POST.get('TP')

LF\_HF= request.POST.get('LF\_HF')

HF\_LF= request.POST.get('HF\_LF')

sampen= request.POST.get('sampen')

higuci= request.POST.get('higuci')

df = pd.read\_csv('Datasets.csv', encoding='latin-1')

def apply\_results(label):

if (label == 'no stress'):

return 0 # No Stress

elif (label == 'stress'):

return 1 # Stress

df['results'] = df['condition'].apply(apply\_results)

x = df["FID"]

y = df["results"]

cv = CountVectorizer(lowercase=False, strip\_accents='unicode', ngram\_range=(1, 1))

FID=FID,

MEAN\_RR=MEAN\_RR,

MEDIAN\_RR=MEDIAN\_RR,

SDRR=SDRR,

RMSSD=RMSSD,

SDSD=SDSD,

SDRR\_RMSSD=SDRR\_RMSSD,

HR=HR,

VLF=VLF,

VLF\_PCT=VLF\_PCT,

LF=LF,

LF\_PCT=LF\_PCT,

LF\_NU=LF\_NU,

HF=HF,

HF\_PCT=HF\_PCT,

HF\_NU=HF\_NU,

TP=TP,

LF\_HF=LF\_HF,

HF\_LF=HF\_LF,

sampen=sampen,

higuci=higuci,

Prediction=val)

**Service provider.py**

rom django.contrib import admin

# Register your models here.

from django.apps import AppConfig

class ResearchSiteConfig(AppConfig):

name = 'Service\_Provider'

from django.db import models

# Create your models here.

response = HttpResponse(content\_type='application/ms-excel')

# decide file name

response['Content-Disposition'] = 'attachment; filename="Predicted\_Data.xls"'

# creating workbook

wb = xlwt.Workbook(encoding='utf-8')

# adding sheet

ws = wb.add\_sheet("sheet1")

# Sheet header, first row

row\_num = 0

font\_style = xlwt.XFStyle()

# headers are bold

font\_style.font.bold = True

# writer = csv.writer(response)

obj = predict\_stress\_detection.objects.all()

data = obj # dummy method to fetch data.

for my\_row in data:

row\_num = row\_num + 1

ws.write(row\_num, 0, my\_row.FID, font\_style)

ws.write(row\_num, 1, my\_row.MEAN\_RR, font\_style)

ws.write(row\_num, 2, my\_row.MEDIAN\_RR, font\_style)

ws.write(row\_num, 3, my\_row.SDRR, font\_style)

ws.write(row\_num, 4, my\_row.RMSSD, font\_style)

ws.write(row\_num, 5, my\_row.SDSD, font\_style)

**CHAPTER-10**

**RESULTS/DISCUSSION**

**10.1SYSTEM TESTING**

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.

**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.

**6.1 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 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.

# 6.2 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.

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

**6.3 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.

**SYSTEM TESTING**

### **TESTING METHODOLOGIES**

The following are the Testing Methodologies:

* **Unit Testing.**
* **Integration Testing.**
* **User Acceptance Testing.**
* **Output Testing.**
* **Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

**1)Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

* The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
* A driver (i.e.) the control program for testing is written to coordinate test case input and output.
* The cluster is tested.
* Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**OTHER TESTING METHODOLOGIES**

**User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

**USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

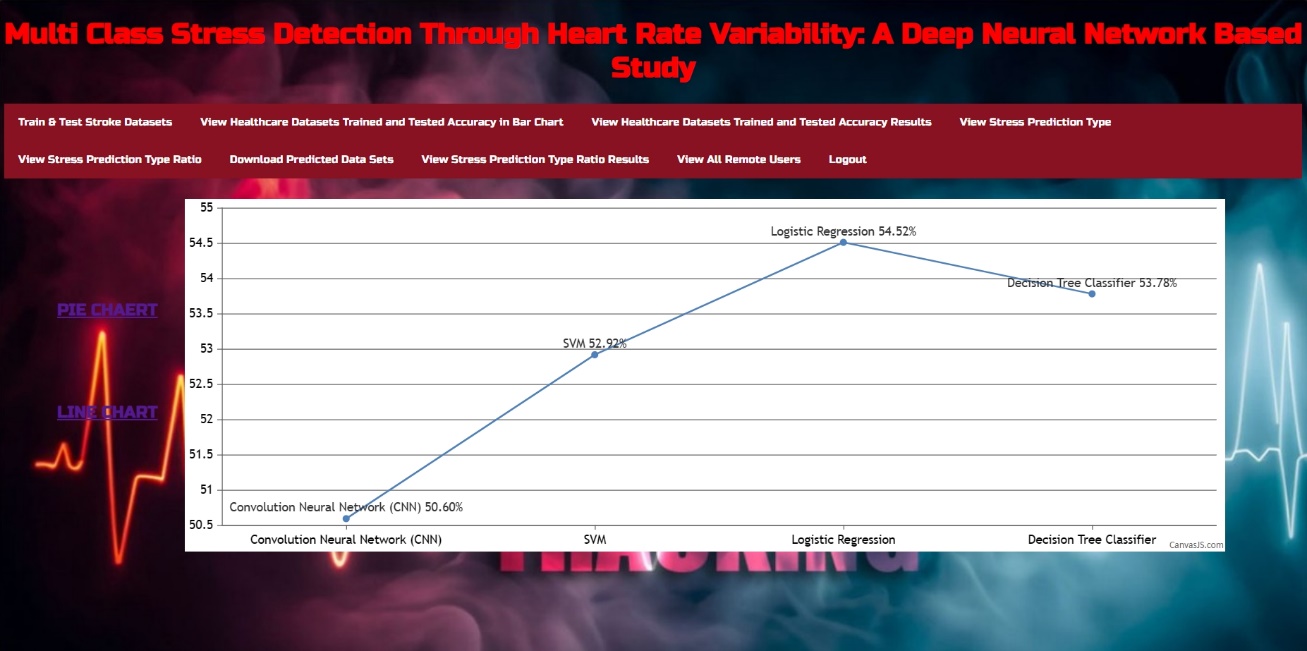
**TESTING STRATEGY :**

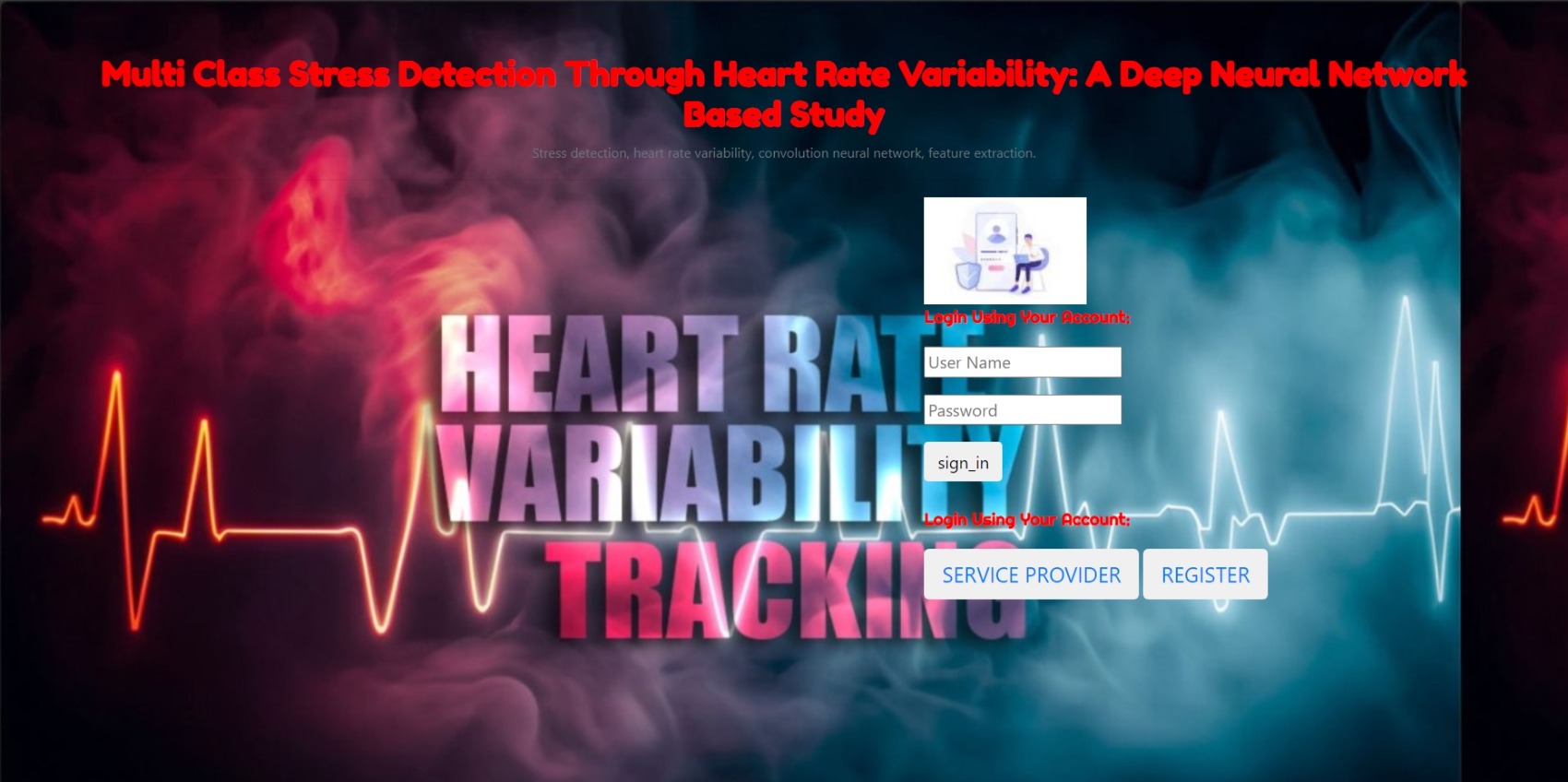
A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strat

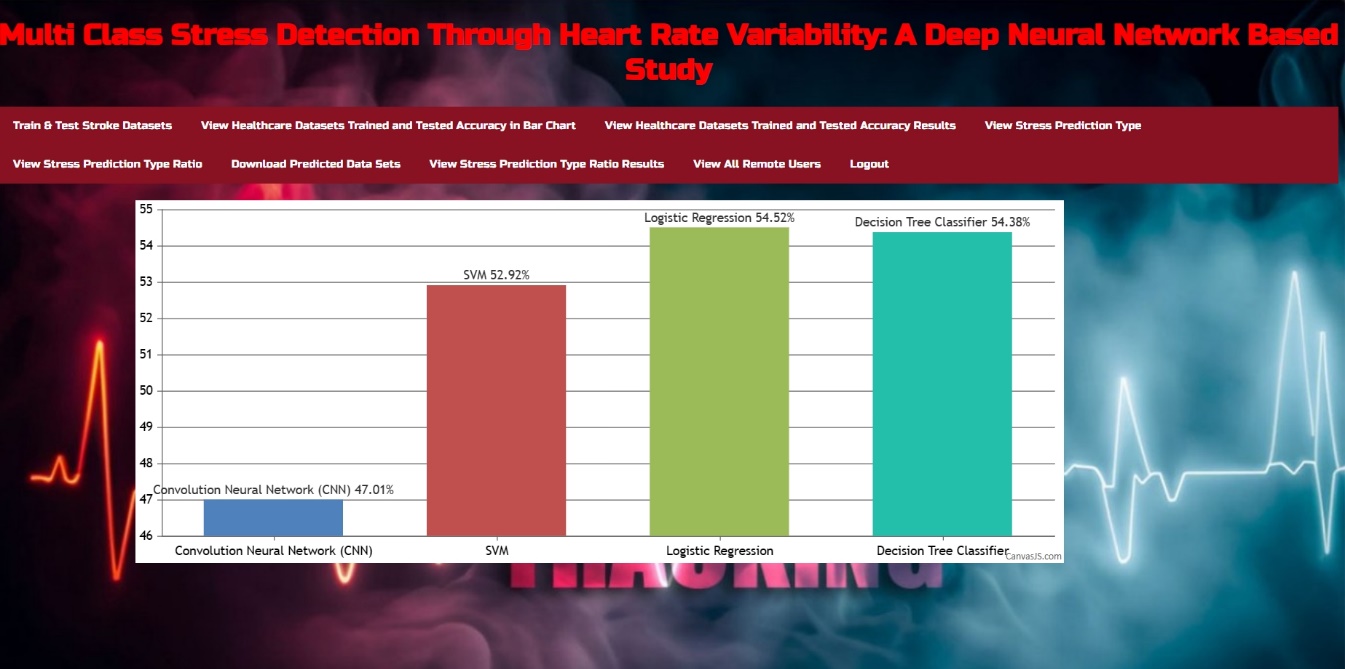
.

**SYSTEM TESTING:**

**UNIT TESTING:**In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules..

**outputscreenshorts**

****

****



**CHAPTER-11**

**CONCLUSION**

In this study, we have developed novel a 1D CNN model for stress level classification using HRV signals and validated the proposed model based on a publicly available dataset, SWELL−KW. In our model, we also applied an ANOVA feature selection technique for dimension reduction. Through extensive training and validation, we demonstrate that our model outperforms the state-of-the-art models in terms of major performance metrics, i.e., Accuracy, Precision, Recall, F1-score, and MCC when all features are employed. Furthermore, our approach with ANOVA feature reduction also achieves excellent performance. For future work, we plan to further investigate the feasibility of optimizing the model to fit it into edge devices so that real-time stress detection can become a reality.

**CHAPTER-12**

**REFERENCES/BIBLIOGRAPHY**

[1] H.-G. Kim, E.-J. Cheon, D.-S. Bai, Y. H. Lee, and B.-H. Koo, ‘‘Stress

and heart rate variability: A meta-analysis and review of the literature,’’

*Psychiatry Invest.*, vol. 15, no. 3, pp. 235–245, Mar. 2018.

[2] D. Muhajir, F. Mahananto, and N. A. Sani, ‘‘Stress level measurements

using heart rate variability analysis on Android based application,’’ *Proc.*

*Comput. Sci.*, vol. 197, pp. 189–197, Jan. 2022.

[3] J. Held, A. Vîslă, C. Wolfer, N. Messerli-Bürgy, and C. Flückiger, ‘‘Heart

rate variability change during a stressful cognitive task in individuals with

anxiety and control participants,’’ *BMC Psychol.*, vol. 9, no. 1, p. 44,

Mar. 2021.

[4] K. M. Dalmeida and G. L. Masala, ‘‘HRV features as viable physiological

markers for stress detection using wearable devices,’’ *Sensors*, vol. 21,

no. 8, p. 2873, Apr. 2021.

[5] J. A. Miranda-Correa, M. K. Abadi, N. Sebe, and I. Patras, ‘‘AMIGOS:

A dataset for affect, personality and mood research on individuals

and groups,’’ *IEEE Trans. Affect. Comput.*, vol. 12, no. 2, pp. 479–493,

Apr./Jun. 2021.

[6] E. Won and Y.-K. Kim, ‘‘Stress, the autonomic nervous system, and

the immune-kynurenine pathway in the etiology of depression,’’ *Current*

*Neuropharmacol.*, vol. 14, no. 7, pp. 665–673, Aug. 2016.

[7] B. Olshansky, H. N. Sabbah, P. J. Hauptman, andW. S. Colucci, ‘‘Parasympathetic

nervous system and heart failure: Pathophysiology and potential

implications for therapy,’’ *Circulation*, vol. 118, no. 8, pp. 863–871,

Aug. 2008.

[8] S. Goel, P. Tomar, and G. Kaur, ‘‘ECG feature extraction for stress recognition

in automobile drivers,’’ *Electron. J. Biol.*, vol. 12, no. 2, pp. 156–165,

Mar. 2016.

[9] V. N. Hegde, R. Deekshit, and P. S. Satyanarayana, ‘‘A review on ECG

signal processing and HRV analysis,’’ *J. Med. Imag. Health Informat.*,

vol. 3, no. 2, pp. 270–279, Jun. 2013.

[10] M. Vollmer, ‘‘A robust, simple and reliable measure of heart rate variability

using relative RR intervals,’’ in *Proc. Comput. Cardiol. Conf. (CinC)*,

Sep. 2015, pp. 609–612.