

A  
Major Project Report  
On  
**Brain Stroke Prediction Using Machine Learning  
Technique**

*Submitted to CMREC, HYDERABAD*

*In Partial Fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**

Submitted  
By

<b>M. Manoj Kumar</b>	<b>(218R1A67A3)</b>
<b>T. Sai Ramya</b>	<b>(218R1A67C5)</b>
<b>R. Narender</b>	<b>(228R5A6711)</b>
<b>S. Shyam Sundar Reddy</b>	<b>(21UJ1A6732)</b>

Under the Esteemed guidance of  
**Mrs.N.Sumanjali**

Assistant Professor, Department of CSE (Data Science)



**Department of Computer Science & Engineering (Data Science)**

**CMR ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501401.

**2024-2025**

# CMR ENGINEERING COLLEGE

## UGC AUTONOMOUS

*(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)*

*Kandlakoya, Medchal Road, Hyderabad-501401*

### Department of Computer Science & Engineering (Data Science)



### CERTIFICATE

This is to certify that the major project entitled “**BRAIN STROKE PREDICTION USING THE MACHINE LEARNING TECHNIQUE**” is a bonafide work carried out by

<b>M. Manoj Kumar</b>	<b>(218R1A67A3)</b>
<b>T. Sai Ramya</b>	<b>(218R1A67C5)</b>
<b>R. Narender</b>	<b>(228R5A6711)</b>
<b>S. Shyam Sundar Reddy</b>	<b>(21UJ1A6732)</b>

in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE) from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision. The results presented in this major project have been verified and are found to be satisfactory. The results embodied in this major project have not been submitted to any other university for the award of any other degree or diploma

<b>Internal Guide</b>	<b>Major Project Coordinator</b>	<b>Head of the Department</b>	<b>External Examiner</b>
<b>Mrs. N. Sumanjali</b> Assistant Professor CSE(Data Science) CMREC	<b>Mr. B. Kumaraswamy</b> Assistant Professor CSE(Data Science), CMREC	<b>Dr. M. Laxmaiah</b> Professor & HOD CSE(Data Science), CMREC	

## **DECLARATION**

This is to certify that the work reported in the present Major project entitled "**Brain Stroke Prediction Using Machine Learning Technique**" is a record of Bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

<b>M. Manoj Kumar</b>	<b>(218R1A67A3)</b>
<b>T. Sai Ramya</b>	<b>(218R1A67C5)</b>
<b>R. Narender</b>	<b>(228R5A6711)</b>
<b>S. Shyam Sundar Reddy</b>	<b>(21UJ1A6732)</b>

## **ACKNOWLEDGMENT**

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Laxmaiah**, HOD, **Department of CSE (Data Science), CMR Engineering College** for their constant support.

We are extremely thankful to Mrs. N.Sumanjali, Assistant Professor, Internal Guide, Department of CSE(DS), for his constant guidance, encouragement, and moral support throughout the project.

We will be failing in duty if We do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mr. B. Kumaraswamy**, Assistant Professor, CSE(DS) Department, Major Project Coordinator for his constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided me for every step.

<b>M. Manoj Kumar</b>	<b>(218R1A67A3)</b>
<b>T. Sai Ramya</b>	<b>(218R1A67C5)</b>
<b>R. Narender</b>	<b>(228R5A6711)</b>
<b>S. Shyam Sundar Reddy</b>	<b>(21UJ1A6732)</b>

# **ABSTRACT**

This project aims to use machine learning to predict stroke risk, a leading cause of long-term disability and mortality worldwide. The study uses a dataset with patient demographic and health features to explore the predictive capabilities of three algorithms: Artificial Neural Networks (ANN), Decision Trees, and Naive Bayes. The primary objectives are to build a predictive model for stroke risk, assess and compare the performance of these algorithms, and deploy the best-performing model in a web-based application for easy access for healthcare professionals and patients. The ANN model demonstrated the highest accuracy among the models, while the Decision Tree model offered interpretability and was useful for clinicians who prioritize transparent decision-making. Naive Bayes was effective in cases where independence between features could be reasonably assumed and performed well due to its simplicity and speed. To implement the models, an interactive web application with a user-friendly interface using Flask was developed. This tool allows users to input individual health data and receive immediate stroke risk predictions, with options to select the prediction model. The system records input and prediction data for future model refinement and personalized health recommendations. Future developments include expanding the model's predictive capacity with additional features and real-time data integration from wearable health monitoring devices. Ensemble learning approaches could also be investigated to further enhance predictive accuracy.

# CONTENTS

TOPIC	PAGE NO
<b>ABSTRACT</b>	
<b>LIST OF FIGURES</b>	
<b>1. INTRODUCTION</b>	
1.1 Overview	1
1.2 Research Motivation	2
1.3 Problem Statement	3
1.4 Applications	3
<b>2. LITERATURE SURVEY</b>	6
<b>3. EXISTING SYSTEM</b>	
3.1 Stroke Risk Prediction Models Using Traditional Statistical	8
3.2 Machine Learning-Based Stroke Prediction Systems	8
3.3 Stroke Prediction Systems Based on Medical Imaging	9
3.4 Mobile Applications and Wearable Devices for Stroke Risk Monitoring	9
<b>4. PROPOSED SYSTEM</b>	
4.1 Overview	11
4.2 Architecture Diagram	11
4.3 Dataflow Diagram	13
<b>5. UML DIAGRAMS</b>	
5.1 Class Diagram	15
5.2 Architecture Diagram	16
5.3 Sequence Diagram	17
5.4 Activity Diagram	19
<b>6. SOFTWARE ENVIRONMENT</b>	
6.1 Programming language and its Advantages and Disadvantages	20
6.2 History of python	24
6.3 Modules used in project	26
6.4 Installation of python	28
6.5 Frontend	35
6.6 Backend	36
<b>7. SYSTEM REQUIREMENTS SPECIFICATIONS</b>	
7.1 Software Requirements	37
7.2 Hardware Requirements	37

<b>8. FUNCTIONAL REQUIREMENTS</b>	
8.1 Output Design and Definition	38
8.2 Input Design, Stages, Types, Media	38
8.3 User Interface	40
8.4 Performance Requirements	41
8.5 Feasibility Study	42
<b>9. METHODOLOGY</b>	
9.1 Requirements Gathering Stage	44
9.2 Analysis Stage	46
9.3 Designing Stage	47
9.4 Development Stage	47
9.5 Integration and Testing Stage	48
9.6 Installation and Acceptance Test Stage	49
9.7 Maintenance	49
<b>10. SYSTEM TESTING</b>	
10.1 Testing	50
10.2 System Testing	50
10.3 Module Testing	50
10.4 Integration Testing	51
10.5 Acceptance Testing	51
<b>11. SOURCE CODE</b>	52
<b>12. RESULTS AND DISCUSSIONS</b>	
12.1 Implementation Description	59
12.2 Results	60
<b>13. CONCLUSION AND REFERENCES</b>	63

## **LIST OF FIGURES**

<b>FIG.NO</b>	<b>DESCRIPTION</b>	<b>PAGE NO</b>
4.1	Architecture Diagram of proposed system	12
4.2	Dataflow Diagram of proposed system	13
5.1	Class Diagram	16
5.2	Use Case Diagram	17
5.3	Sequence Diagram	18
5.4	Activity Diagram	19
6	Python installation Diagrams	29
9.1	Umbrella Model	44
9.2	Requirements Gathering	45
9.3	Analysis Stage	46
9.4	Designing Stage	47
9.5	Coding Stage	48
9.6	Integration and Testing Stage	48
9.7	Installation	49
12.1	Home Page	60
12.2	User Login Page	61
12.3	Prediction Page	61
12.4	Result Page	62
12.5	Result page	62



# 1. Introduction

## 1.1 Overview

Stroke is a major global health concern, affecting millions and burdening healthcare systems. Early detection and prediction of stroke risk are crucial for timely medical intervention, potentially saving lives and preventing severe disabilities. Machine learning (ML) has emerged as a promising tool in healthcare, offering robust support in diagnosis, prognosis, and treatment planning. This project focuses on applying ML techniques to predict stroke risk, enabling healthcare providers and patients to take preventative measures proactively. The study explores the effectiveness of different ML algorithms—Artificial Neural Networks (ANN), Decision Trees, and Naive Bayes—in accurately predicting the likelihood of stroke based on patient demographic and medical data. A comprehensive dataset with features related to known stroke risk factors was used, and data preprocessing involved handling missing values, encoding categorical variables, and normalizing numerical features. The three ML models were trained and validated on a labeled dataset, with ANN showing superior accuracy due to its layered structure and deep learning capabilities, Decision Tree's transparent structure making it easier to interpret, and Naive Bayes providing reliable predictions and computational efficiency. Decision Trees are highly beneficial for machine learning tasks due to their simplicity and interpretability. They provide a clear visualization of the decision-making process, making them easy to understand and explain even to non-technical stakeholders. They effectively handle both classification and regression problems, capturing complex, nonlinear relationships in the data. Decision Trees do not require feature scaling or normalization and can work with a mix of categorical and numerical data. They highlight feature importance, enabling insights into which variables contribute most to predictions. Additionally, they are computationally efficient, robust to outliers, and can handle irrelevant features by focusing on significant splits.

Naive Bayes is a probabilistic machine learning algorithm based on Bayes' Theorem, widely used for classification tasks. It assumes that all features are conditionally independent, which simplifies computations and makes the model efficient, even with high-dimensional data. Naive Bayes calculates the probability of a class given the input features by combining prior probabilities, the likelihood of features within each class, and the overall feature probability. Despite its "naive" assumption of independence, it performs remarkably well for problems like spam detection, sentiment analysis, and medical diagnoses.

## 1.2 Research Motivation

The increasing global burden of stroke, a leading cause of mortality and long-term disability, necessitates the development of accurate and early risk prediction methods. Traditional stroke risk assessment relies on clinical evaluations and statistical models, which may not fully capture the complex interactions between various physiological and lifestyle-related risk factors. Recent advancements in artificial intelligence (AI) and machine learning (ML) offer a promising alternative by enabling data-driven, high-precision predictive models. Several studies have demonstrated that ML algorithms can identify subtle patterns in medical data that might be overlooked by conventional approaches. By leveraging large datasets containing demographic, clinical, and lifestyle variables, ML models can improve stroke risk prediction, allowing for timely medical interventions. Research has shown that models such as Artificial Neural Networks (ANN), Decision Trees, and Naïve Bayes can effectively analyze stroke-related risk factors, leading to significant improvements in predictive accuracy. The motivation behind this project stems from the need to develop a robust, automated, and efficient stroke prediction system that enhances early detection capabilities. By integrating ML techniques with electronic health records (EHRs) and other medical databases, this project aims to facilitate proactive stroke prevention strategies, reducing the overall healthcare burden.

Furthermore, the widespread availability of digital health technologies, such as wearable devices and mobile health applications, provides an opportunity to incorporate ML-driven stroke risk assessments into everyday healthcare monitoring. This accessibility is particularly valuable in remote or underserved regions, where access to specialized healthcare professionals may be limited. By utilizing ML for stroke prediction, healthcare providers and individuals can benefit from early warnings, allowing for lifestyle modifications and timely medical interventions.

In summary, this project is motivated by the potential of ML to revolutionize stroke prediction through its ability to analyze complex medical data with high accuracy. By leveraging ML algorithms, we aim to develop a non-invasive, efficient, and widely accessible tool for stroke risk assessment, ultimately contributing to improved patient outcomes and global public health efforts.

### 1.3 Problem Statement

Existing brain health monitoring systems primarily rely on conventional clinical assessments or imaging techniques such as magnetic resonance imaging (MRI) and computed tomography (CT) scans. While these methods are highly effective in diagnosing stroke after occurrence, they pose challenges that hinder their widespread adoption for proactive and continuous stroke risk assessment. MRI and CT scans provide detailed insights into brain structure and vascular conditions; however, they are expensive, time-consuming, and require specialized medical facilities. This makes them impractical for routine monitoring, especially for individuals at high risk of stroke who may require frequent assessments. Additionally, these imaging techniques are reactive rather than preventive, detecting stroke after significant damage has already occurred. Traditional risk assessment models rely on patient history, clinical biomarkers, and manual evaluation, which may not capture subtle patterns indicative of an impending stroke. Moreover, these methods depend heavily on periodic clinical visits, leading to delays in early detection and intervention. Machine learning-based stroke prediction models offer a promising alternative by leveraging vast amounts of patient data, including demographics, medical history, lifestyle factors, and physiological parameters. By analyzing complex patterns and correlations, these models can provide early warnings of stroke risk, enabling timely preventive measures and reducing the likelihood of severe outcomes. However, challenges such as data availability, model interpretability, and integration into real-world healthcare systems must be addressed to maximize their effectiveness and reliability.

### 1.4 Applications

Brain stroke prediction using machine learning offers a wide range of applications across healthcare, research, and technology-driven solutions. By leveraging predictive analytics and pattern recognition, these systems enable early intervention and improved patient outcomes. The key applications include:

#### 1. Personal Health Monitoring

- **Early Stroke Risk Assessment:** Machine learning models can analyze lifestyle, genetic, and health data to provide individuals with early warnings about potential stroke risks.
- **Health Trend Analysis:** Users can track their risk factors over time, enabling them to take preventive measures through lifestyle modifications.
- **Wearable Device Integration:** Smartwatches and health-monitoring devices can

integrate predictive models to alert users about irregular health parameters related to stroke risk.

## **2. Clinical Applications**

- **Remote Patient Monitoring:** Healthcare professionals can monitor at-risk patients remotely using machine learning-driven predictive tools, reducing the need for frequent hospital visits.
- **Emergency Stroke Prediction:** AI models can analyze real-time health data from emergency response systems to predict and respond to stroke symptoms faster.
- **Personalized Treatment Plans:** By assessing an individual's risk factors, machine learning can help doctors create personalized preventive care strategies.

## **3. Mental Health Assessment**

- **Neurodegenerative Disease Monitoring:** Stroke risk is linked to conditions like dementia and Alzheimer's. Predictive models can assist in monitoring cognitive health deterioration.
- **Mood and Stress Analysis:** Machine learning can assess physiological and behavioral indicators that contribute to stroke risks due to chronic stress or depression.

## **4. Workplace Wellness Programs**

- **Employee Health Monitoring:** Organizations can integrate stroke risk prediction models into wellness programs to encourage preventive healthcare among employees.
- **Occupational Health Assessments:** High-risk professionals, such as those in stressful or physically demanding jobs, can benefit from stroke risk monitoring as part of workplace safety protocols.

## **5. Research and Data Collection**

- **Large-Scale Health Studies:** Machine learning models can process vast datasets to uncover new stroke risk factors and improve medical understanding.
- **Predictive Epidemiology:** Public health organizations can use AI-driven analysis to predict stroke prevalence in different populations, enabling better resource allocation.

## **6. Integration with Other Technologies**

- **Smart Home Systems:** AI-powered stroke prediction can integrate with smart

home devices to alert caregivers or emergency services if a high-risk event is detected.

- **Electronic Health Records (EHR) Integration:** Predictive models can be embedded in healthcare management systems to enhance decision-making for stroke prevention.

## **7. Educational Tools**

- **Public Awareness Campaigns:** AI-powered stroke risk calculators can help educate individuals about risk factors and encourage preventive healthcare.
- **Medical Training and Decision Support:** Healthcare professionals can use machine learning models to study stroke risk factors and improve clinical decision-making.

## **8. Insurance and Wellness Programs**

- **Health Insurance Optimization:** Insurers can offer personalized plans based on machine learning-driven stroke risk assessments, incentivizing preventive healthcare.
- **Corporate Health Initiatives:** Businesses can implement AI-driven health monitoring as part of corporate wellness programs to reduce long-term healthcare costs.

## 2. Literature Survey

[1] Patel R, al. Proposed a Time Stroke Prediction Model with Federated Learning System. To protect patient privacy while enabling accurate stroke predictions across multiple healthcare institutions. By allowing local models at each site to contribute to a global model without sharing raw patient data, this approach enhances prediction accuracy while maintaining compliance with privacy regulations. The model is trained on data from over 10,000 patients across various demographics and achieves an accuracy rate of 95%. The study highlights the model's effectiveness in providing timely predictions to aid in preventive stroke treatment.

[2] Zhang H, et al. Proposed a Explainable AI for Stroke Risk Assessment Using Gradient-Boosted Decision Trees in the year 2024. This research introduces an explainable AI (XAI) model using Gradient-Boosted Decision Trees (GBDT) for stroke prediction, focusing on interpretability and transparency. The XAI approach allows healthcare professionals to understand the influence of each factor on stroke risk. Key predictive features identified include age, hypertension, and blood glucose levels, among others. With an accuracy of 92%, this model aims to provide clinicians with a clear and interpretable framework to predict stroke risk and guide patient counseling on modifiable risk factors.

[3] Patel S, et al. Proposed a Predicting Stroke Risk with Hybrid Ensemble Models in the year 2023. This study develops a hybrid ensemble model combining Decision Trees, Gradient Boosting, and k-Nearest Neighbors to improve stroke prediction accuracy. The hybrid approach capitalizes on the strengths of each model, achieving a balanced accuracy rate of 93%. The model was trained on a large dataset with multiple stroke risk factors, including age, hypertension, and lifestyle habits. By leveraging ensemble methods, this model demonstrates robust performance, even with noisy data, and provides reliable predictions for clinical decision support.

[4] M. Anand Kumar and N. Abiram proposed a Stroke Disease Prediction based on ECG Signals using Deep Learning Techniques in the year 2023. Stroke-related diseases are rapidly increasing day by day due to the changes in environmental factors including lifestyles, food habits, and stress-related working cultures. According to a recent report from World Health Organization (WHO), Stroke is the second largest disease after

cardiovascular disease that leads to death. Early diagnosis of stroke-related diseases was one of the major requirements for patients as well as medical professionals. This work proposed a framework based on Long Short-term Memory (LSTM) network for predicting stroke-related diseases with ECG data and other parameters. The experimental results show that 90% accuracy results with the combination of ECG data and the Deep learning approach.

[5] Rajasekaran S, et al. Proposed a Stroke Prediction Using Machine Learning Models. A Comparative Analysis in the year 2021. This study compares various machine learning algorithms, including Logistic Regression, Random Forest, and Support Vector Machines (SVM), for stroke prediction. The study aims to identify the most effective algorithm for predicting stroke risk based on commonly available clinical and demographic data. With an accuracy of 91%, Random Forest performed best among the models tested, followed closely by SVM. This research provides valuable insights into how different machine learning algorithms handle stroke prediction tasks.

### 3. EXISTING SYSTEM

In the field of stroke prediction, several systems and methodologies have been developed to aid healthcare professionals in identifying individuals at risk and making timely interventions. These existing systems leverage various machine learning (ML) algorithms, data sources, and diagnostic tools to predict stroke risk and assist in clinical decision-making. Below are some of the prominent systems used for stroke prediction.

**3.1 Stroke Risk Prediction Models Using Traditional Statistical:** Historically, many stroke prediction systems have relied on traditional statistical methods to predict stroke risk based on patient data. These models use risk factors such as age, gender, blood pressure, cholesterol levels, smoking status, and family medical history to calculate a patient's likelihood of experiencing a stroke. Some of the widely used models include

**Framingham Stroke Risk Profile (FSRP):** This is one of the most well-known stroke risk assessment tools. The FSRP calculates a patient's 10-year stroke risk based on factors like age, gender, blood pressure, smoking status, diabetes, and cholesterol levels. However, its predictions are based on statistical relationships rather than machine learning models, and its accuracy may be limited when applied to diverse populations or novel data.

**QRISK:** Another commonly used risk assessment tool, QRISK, predicts the likelihood of cardiovascular events, including stroke, by analyzing factors like age, smoking, body mass index (BMI), and medical history. QRISK is widely used in clinical practice but is also based on traditional statistical methods rather than modern machine learning approaches.

**3.2 Machine Learning-Based Stroke Prediction Systems:** As machine learning has evolved, more advanced systems have been developed that use algorithms to analyze larger and more complex datasets, improving prediction accuracy and decision-making.

**Random Forest, Decision Tree, and Support Vector Machines (SVM):** These models have been successfully used for stroke prediction by learning from historical medical data. They are often trained on datasets that include patient demographics, clinical features, and lifestyle factors. These models can automatically detect complex patterns in the data, leading to more accurate predictions compared to traditional statistical models.

**Artificial Neural Networks (ANN):** ANN is one of the most powerful and widely used



machine learning algorithms in medical applications. It can handle nonlinear relationships between input features and provide higher accuracy in stroke prediction. ANN models are used in systems that take into account various risk factors such as age, hypertension, cholesterol levels, and smoking to predict stroke probability. These systems can also learn from new data and improve predictions over time.

**Logistic Regression:** Logistic regression models are another widely used technique in stroke prediction, especially in clinical settings. These models predict the likelihood of stroke based on input features like blood pressure, heart rate, glucose levels, and lifestyle choices. Although simpler than ANN, logistic regression models are still effective for predicting stroke risk in certain contexts.

**3.2 Stroke Prediction Systems Based on Medical Imaging:** Medical imaging plays a crucial role in the diagnosis of strokes, and with the advancements in artificial intelligence (AI) and deep learning, new systems have been developed to assist healthcare providers in detecting stroke from images like CT scans, MRIs, and angiograms. These systems use convolutional neural networks (CNNs) and other deep learning algorithms to analyze medical images for early signs of stroke.

**Deep Learning for Image Analysis:** CNN-based models have been trained to detect stroke-related abnormalities such as ischemic stroke, hemorrhagic stroke, and cerebral edema from medical imaging. These systems can automatically identify subtle changes in brain images that may indicate early signs of a stroke, helping radiologists and doctors make quicker, more accurate diagnoses.

**Stroke Detection Using CT and MRI:** Systems like BrainCT and BrainMRI are being used to analyze CT and MRI scans for stroke detection. These systems use deep learning techniques to segment brain tissues, detect blockages or hemorrhages, and provide a diagnosis, all in a fraction of the time it would take a radiologist to do manually.

### **3.4 Mobile Applications and Wearable Devices for Stroke Risk Monitoring**

The rise of mobile technology and wearable health devices has enabled continuous monitoring of individuals' health status. Several stroke prediction systems are now integrated into mobile apps and wearables that track real-time health data, including heart rate, blood pressure, glucose levels, and activity patterns.

**Wearables and Smartwatches:** Devices like the Apple Watch, Fitbit, and other smartwatches track user health metrics continuously. These devices use machine learning algorithms to monitor factors such as heart rate variability, irregular heart rhythms (like atrial fibrillation), and blood pressure. If these devices detect any abnormal readings, they

can alert users to potential stroke risks, prompting them to seek medical attention.

## Drawbacks

**Signal Complexity:** SCG signals are formed from multiple overlapping mechanical events of the heart, complicating the extraction of meaningful stroke-related features.

**Technological Limitations:** Older SCG sensors lacked the resolution for precise measurements. Despite improvements, these limitations delayed SCG's integration into stroke prediction systems.

**Motion Artifacts:** SCG is highly sensitive to respiratory and body motion artifacts, which can obscure the heart's mechanical signals and affect prediction outcomes.

**Lack of Standardization:** The absence of a universally accepted SCG acquisition and analysis protocol leads to inconsistent performance and results across different models and studies.

**Limited Clinical Validation:** SCG-based stroke risk systems still lack robust clinical backing, with few large-scale validations comparing them to established tools like ECG or MRI.

**Data Quality and Availability:** ML models require large, clean, and diverse datasets. Many healthcare datasets are incomplete, biased, or inconsistently labeled. Privacy concerns and limited access to medical records hinder data collection.

**Complexity and Interpretability:** Many ML models (e.g., deep learning) operate as “black boxes”. Clinicians often struggle to trust or understand decisions without transparent reasoning.

**Integration with Existing Healthcare Systems:** Technical challenges exist in embedding ML tools into EHRs (Electronic Health Records). Workflow disruptions may occur if integration is not seamless. Compatibility with various hospital systems is inconsistent.

**Over-reliance on Technology:** Dependence on AI may reduce clinical judgment or oversight. Technology may fail in unexpected scenarios or edge cases. Critical thinking and human validation are still essential.

**Generalization to New Data:** Models trained on one population may not generalize to others (e.g., ethnic, geographic, or demographic differences). Continuous re-training with new data is needed to maintain performance.

**User Acceptance and Trust:** Clinicians and patients may be skeptical of AI-driven decisions. Adoption depends on trust in the model, its accuracy, and transparency. Training and education are needed to build confidence.

**Limited Real-Time Prediction:** Some models are not optimized for real-time or emergency settings. Computational demands may slow down predictions. Real-time integration into hospital systems remains challenging.

## 4. PROPOSED METHODOLOGY

### 4.1 Overview

The proposed system aims to address the limitations and challenges of existing stroke prediction models by integrating advanced machine learning techniques, enhancing data quality, improving interpretability, and ensuring real-time prediction capabilities. The system leverages multiple machine learning algorithms—Artificial Neural Networks (ANN), Decision Trees, and Naive Bayes—along with a user-friendly web interface for healthcare professionals and patients. The core of the proposed system is the integration of three distinct machine learning algorithms: Artificial Neural Networks (ANN), Decision Trees, and Naive Bayes classifiers. Each algorithm is selected based on its unique strengths to collectively improve the robustness and versatility of the system. The system also includes a user-friendly web-based interface that allows healthcare professionals and patients to interact with the system in a seamless manner. This interface supports data entry, displays risk assessment results, and provides real-time feedback based on the analysis of input data. To ensure accurate predictions, the system processes patient data such as age, blood pressure, glucose levels, body mass index (BMI), smoking status, heart disease, and other lifestyle or clinical factors. A preprocessing pipeline is implemented to handle noise, missing values, and normalization, which significantly contributes to data quality improvement. Moreover, the system is capable of making real-time predictions, thanks to its optimized architecture and efficient algorithmic backend. The prediction results are categorized into different risk levels (e.g., Low, Medium, High), enabling timely medical interventions and personalized patient monitoring.

In summary, the proposed system combines the strengths of multiple machine learning techniques with an intuitive user interface to form a comprehensive solution for brain stroke prediction. It holds the potential to transform the conventional stroke risk assessment process by offering a more accurate, interpretable, and accessible tool for use in modern healthcare environments.

**4.2 Architecture Diagram:** 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. System

architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

**Various organizations define systems architecture in different ways, including:**

An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.

Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.

If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and a plan for upgrading and/or replacing dated equipment and software.

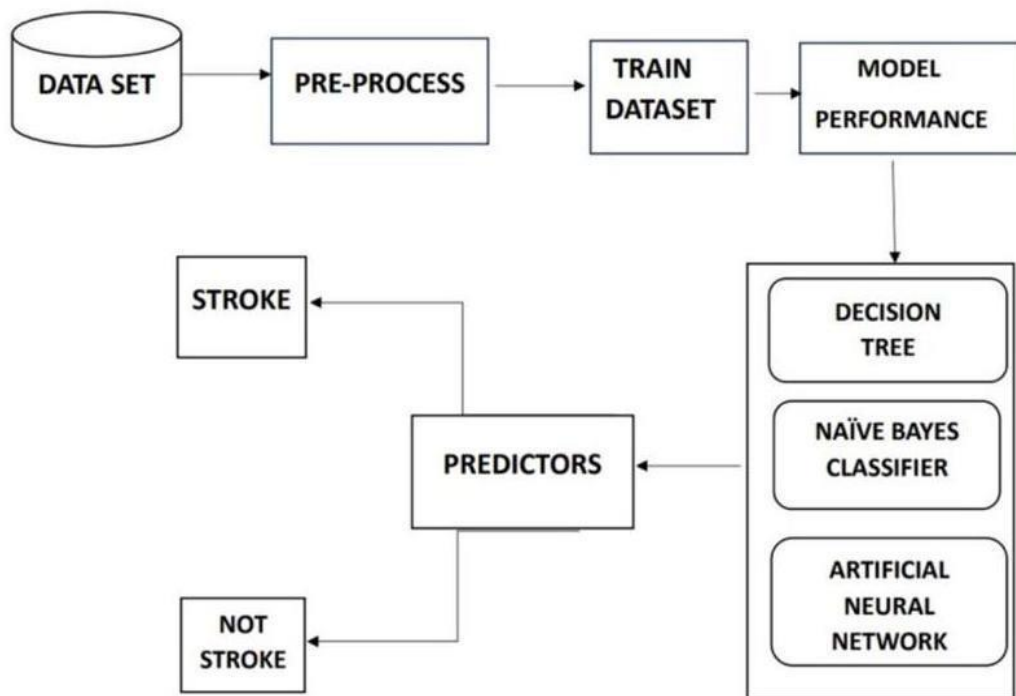


Fig.no.4.1 Architecture Diagram

## 4.3 Dataflow Diagram

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

### LEVEL 0

This stage is to create the Level 0 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.

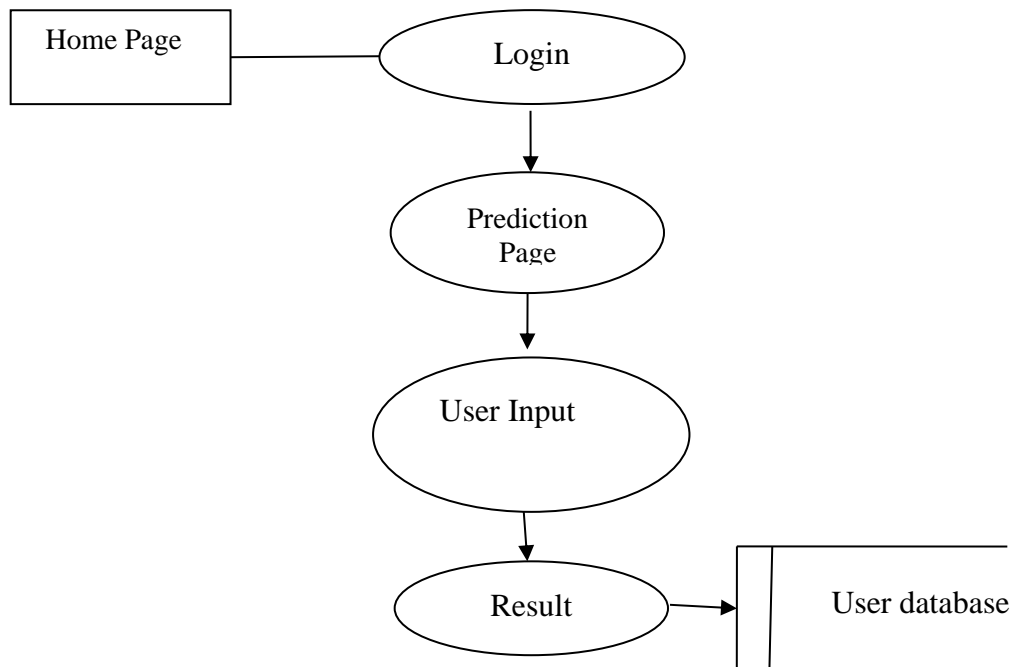


Fig .no .4.2 Dataflow Diagram

## **Advantages**

### **1.Enhanced Prediction Accuracy:**

- ML models can identify complex and nonlinear relationships in data.
- Improves the precision of stroke risk classification over traditional statistical methods.

### **2.Real-Time Risk Assessment:**

- Once trained, ML models provide instant predictions.
- Enables timely medical interventions and faster clinical decision-making.

### **3.Model Interpretability and Transparency:**

- Algorithms like DecisionTrees and Naive Bayes offer understandable outputs.
- Helps healthcare professionals trust and validate the system's recommendations.

### **4.Continuous Improvement and Adaptation:**

- Enhances adaptability across different populations and improves model performance over time.

### **5.Efficient Processing of Large Datasets:**

- Capable of handling and learning from vast amounts of medical and lifestyle data.
- . Increases the robustness of predictions.

## 5. UML DAIGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

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

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Integrate best practices.
- **Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

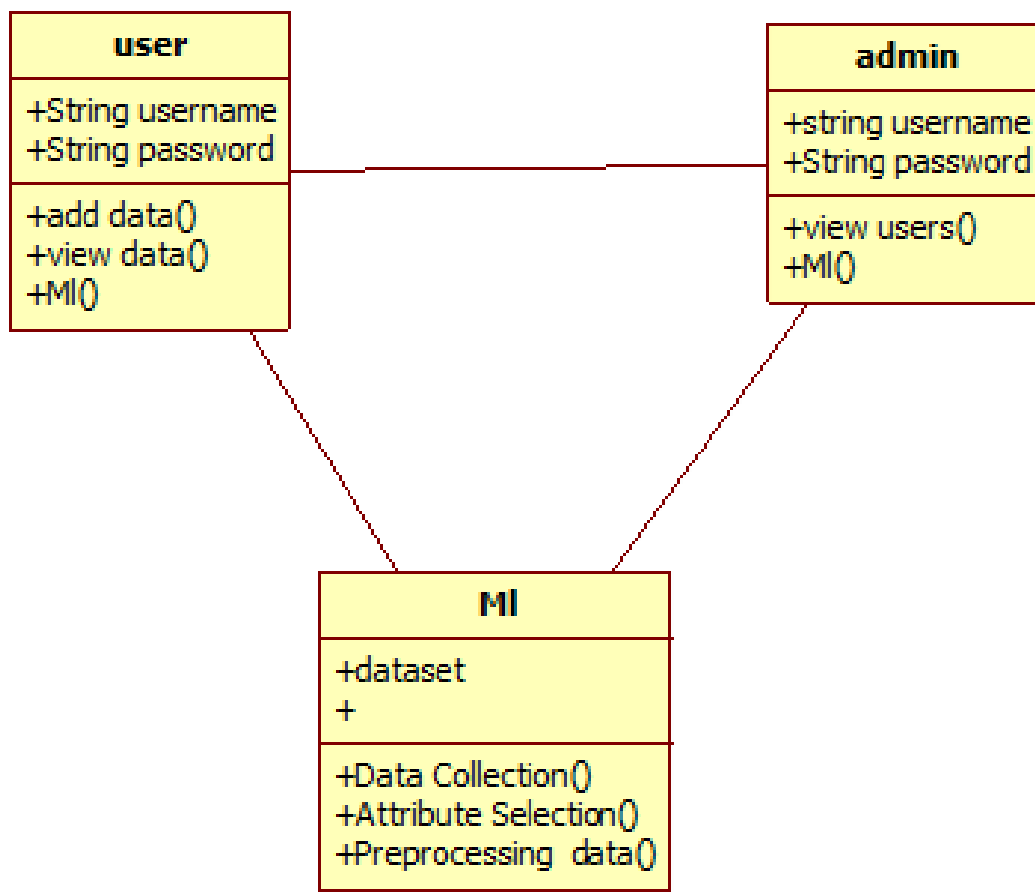


Fig.no.5.1 Class Diagram

- **Use case Diagram**

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



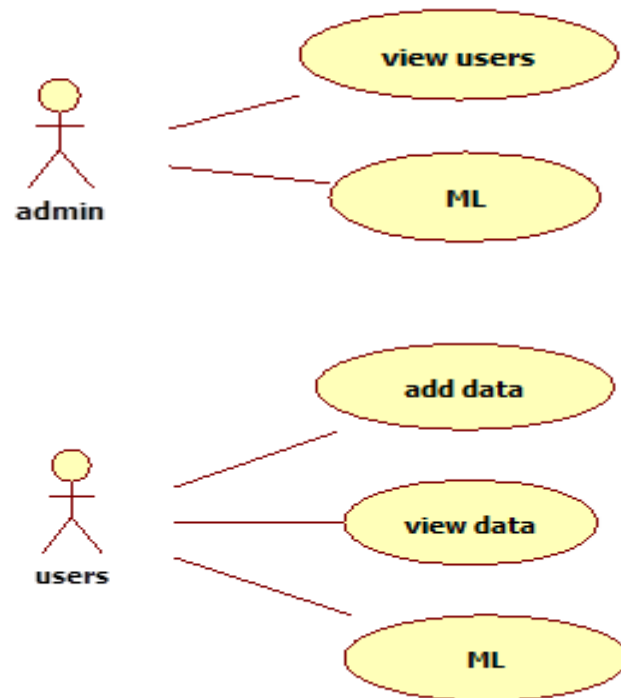


Fig. no. 5.2 Use case Diagram

- **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. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

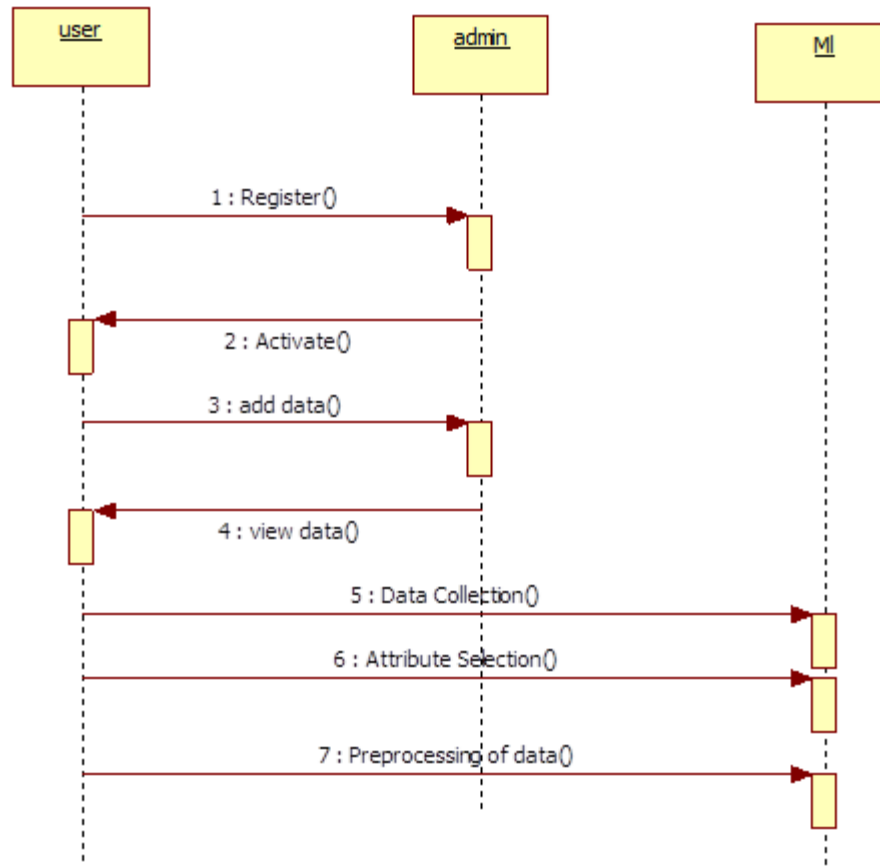


Fig.no.5.3SequenceDiagram

- **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.

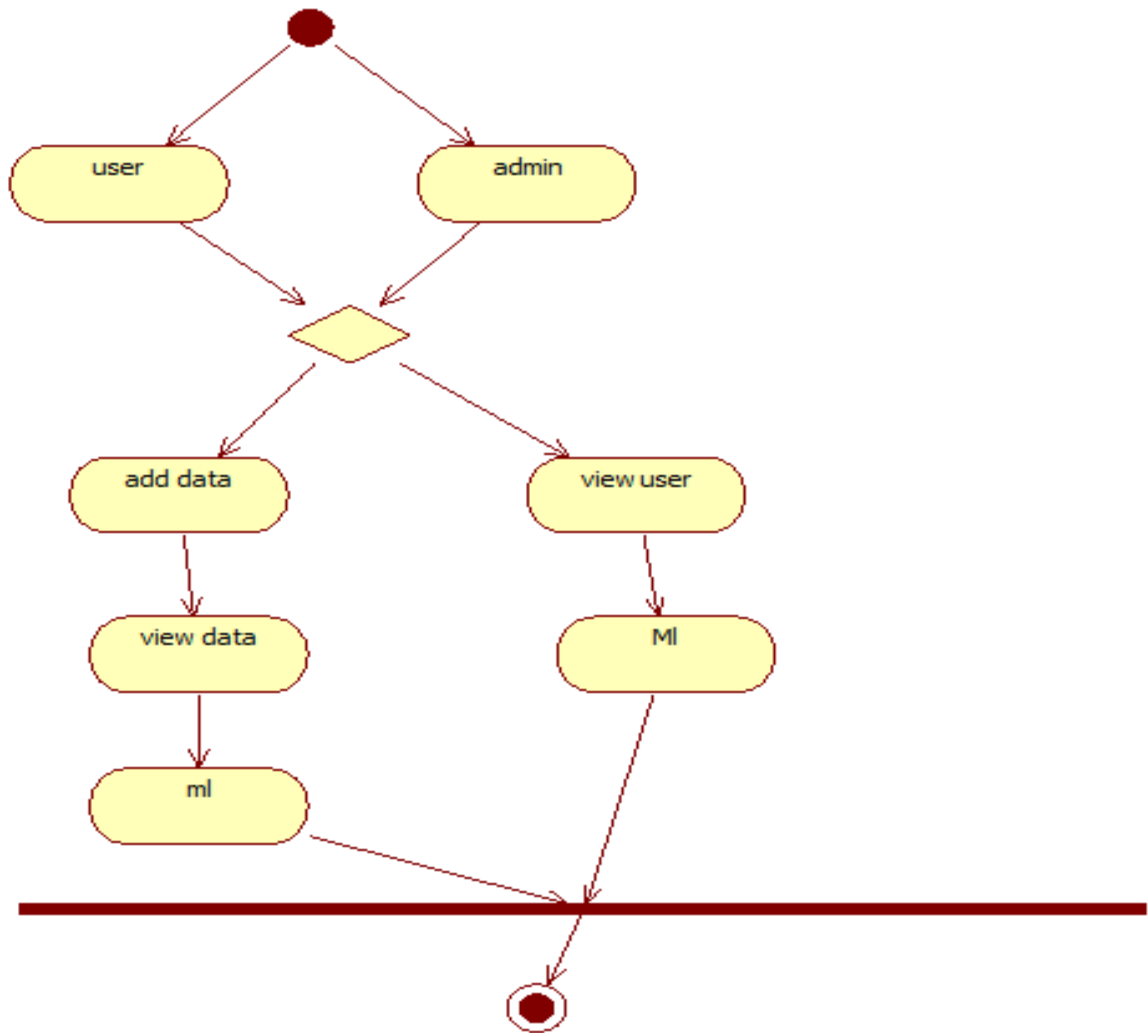


Fig.no.5.4 Activity Diagram

## **6. SOFTWARE ENVIRONMENT**

### **6.1 Python Programing Language**

#### **What is Python?**

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

#### **Advantages of Python**

Let's see how Python dominates over other languages.

##### **1. Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading,

databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

## **2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

## **3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

## **4. Improved Productivity**

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

## **5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

## **6. Simple and Easy**

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

## **7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code

## **8. Object- Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

## **9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

## **10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

## **11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

## **Advantages of Python Over Other Languages**

### **1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

### **2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

### **3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

### **Disadvantages of Python**

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

#### **1. Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

#### **2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonselle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

#### **3. Design Restrictions**

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

#### **4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java Data Base Connectivity) and ODBC (Open Data Base Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

## 5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

## 6.2 History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners<sup>1</sup>, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin- end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

### Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map,



filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced.

This release included list comprehensions, a full garbage collector and it was supporting

unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it. Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behavior.
- Text Vs. Data Instead of Unicode Vs. 8-bit

## **Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers— even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

## **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

## **6.3 Modules**

### **TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

### **NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions

- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

## **Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

## **Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

## **Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

## **6.4 Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows

### **How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is

how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices. Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

### Download the Correct version into the system

**Step 1:** Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.








**Step 2:** Click on the Download Tab.



**Step 3:** You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	<a href="#">Release Notes</a>
Python 3.6.9	July 2, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.3	March 25, 2019	 Download	<a href="#">Release Notes</a>
Python 3.4.10	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 3.5.7	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 2.7.16	March 4, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.2	Dec. 24, 2018	 Download	<a href="#">Release Notes</a>

**Step 4:** Scroll down the page until you find the Files option.

**Step 5:** Here you see a different version of python along with the operating system.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPU
<a href="#">Unipped source tarball</a>	Source release		68111671e5b2db4ae7709ab1b7079be	23017663	3x6
<a href="#">XZ compressed source tarball</a>	Source release		d33e4aee6097051c3eca45ee3004803	17131432	3x6
<a href="#">macOS 64-bit/32-bit installer</a>	Mac OS X	for Mac OS X 10.5 and later	6428b4fa75c3daf1a4c2c8a1ce08e6	34898416	3x6
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later	5dd607c30217a45773b5e4b36b243f	20032845	3x6
<a href="#">Windows help file</a>	Windows		0639995734b2a682ac58ade0b4f7cd2	8131761	3x6
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	9800c3c7629e0b0a0e02184a0729a2	7504391	3x6
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	a7023e0ba076d5b0b30c1a163e563400	2688368	3x6
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	28c31c908b0d73ae0653a3b031b4b02	1362904	3x6
<a href="#">Windows x86 embeddable zip file</a>	Windows		9fab30d18043179fda94112574129d0	6741626	3x6
<a href="#">Windows x86 executable installer</a>	Windows		33c002942a5444a3d04c1476394789	25663848	3x6
<a href="#">Windows x86 web-based installer</a>	Windows		2b670cf0d117d83c30983ca371007c	1134600	3x6

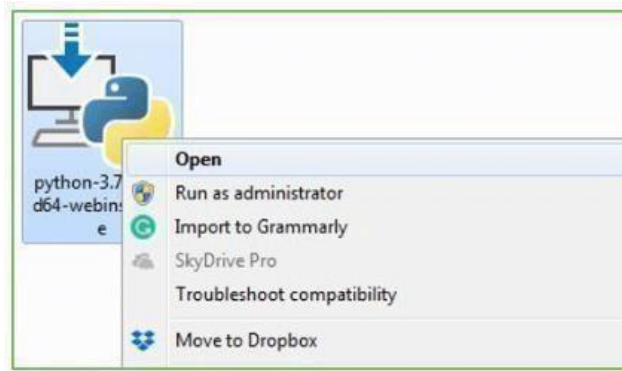
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

### Installation of Python

**Step 1:** Go to Download and Open the downloaded python version to carry out the installation process.



**Step 2:** Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



**Step 3:** Click on Install NOW After the installation is successful. Click on Close.



With these above three steps on python installation, you have successfully and

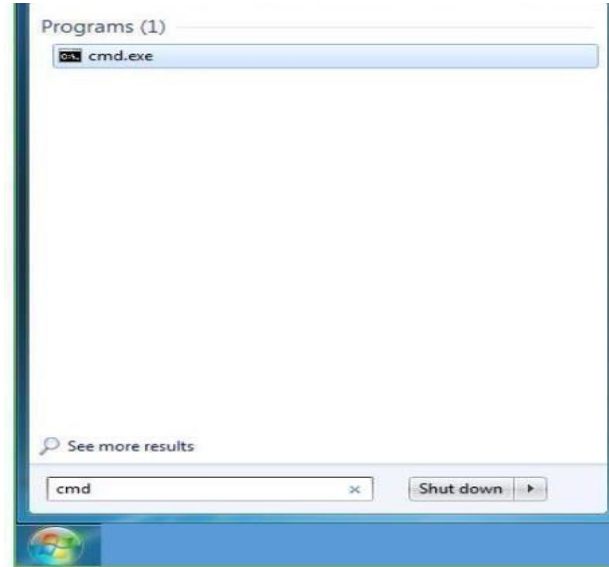


correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes. Verify the Python Installation

**Step 1:** Click on Start

**Step 2:** In the Windows Run Command, type “cmd”.



**Step 3:** Open the Command prompt option.

**Step 4:** Let us test whether the python is correctly installed. Type python -V and press Enter.



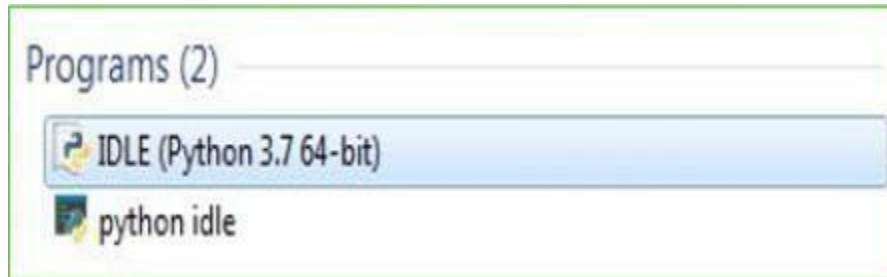
**Step 5:** You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

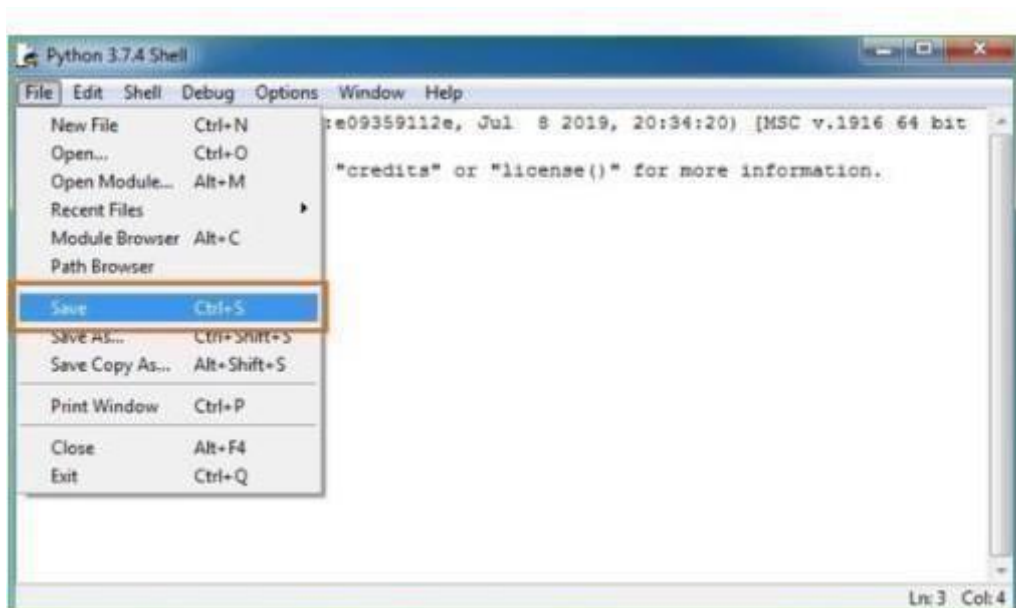
**Step 1:** Click on Start

**Step 2:** In the Windows Run command, type “python idle”.



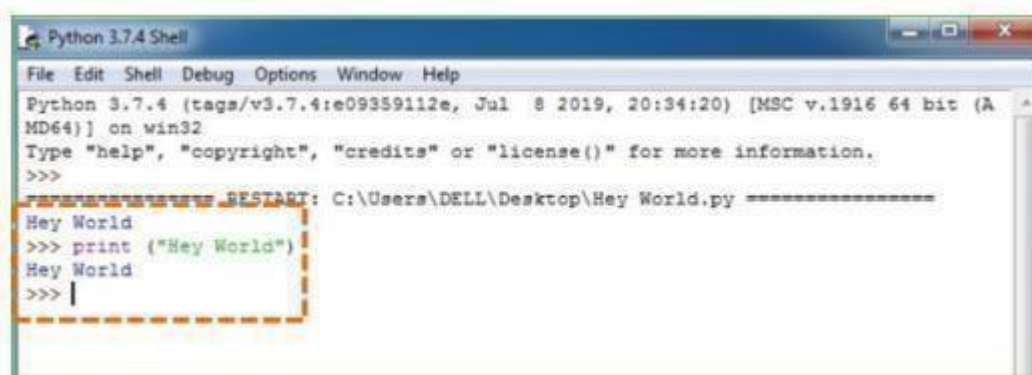
**Step 3:** Click on IDLE (Python 3.7 64-bit) and launch the program

**Step 4:** To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



**Step 5:** Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

**Step 6:** Now for e.g., enter print (“Hey World”) and Press Enter.

A screenshot of a Python 3.7.4 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The title bar says 'Python 3.7.4 Shell'. The main text area shows the following content: 'Python 3.7.4 (tags/v3.7.4:09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', '>>>', '\*\*\*\*\* RESTART: C:\Users\DELL\Desktop\Hey World.py \*\*\*\*\*', 'Hey World', '>>> print ("Hey World")', 'Hey World', '>>> |'. A dashed orange box highlights the output 'Hey World' and the command '>>> print ("Hey World")'.

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

## 6.5 FRONTEND:

### 1. Hypertext Markup Language(HTML)

**HTML** (Hypertext Markup Language) is a text-based approach to describing how content contained within an HTML file is structured. This markup tells a web browser how to display text, images and other forms of multimedia on a webpage. HTML is a formal recommendation by the World Wide Web Consortium (W3C) and is generally adhered to by all major web browsers, including both desktop and mobile web browsers.

**2. HTML Work** **HTML** is a text file containing specific syntax, file and naming conventions that show the computer and the web server that it is in HTML and should be read as such. By applying these HTML conventions to a text file in virtually any text editor, a user can write and design a basic webpage, and then upload it to the internet. The most basic of HTML conventions is the inclusion of a document type declaration at the beginning of the text file. This always comes first in the document, because it is the piece that affirmatively informs a computer that this is an HTML file. The document header typically looks like this: `<!DOCTYPE html>`. It should always be written that way, without any content inside it or breaking it up. Any content that comes before this declaration will not be recognized as HTML by a computer. Doctypes are not just used for HTML, they can apply to the creation of any document that uses SGML (Standard Generalized Markup Language).

**2. Cascading Style Sheets (CSS)** CSS, or Cascading Style Sheets, is a fundamental technology in web development that defines the presentation and layout of HTML documents. Serving as a style language, CSS enables the separation of content from design, allowing developers to control the appearance of web pages consistently across various devices and screen sizes. The working process involves selecting HTML elements and applying style rules to define attributes like colors, fonts, spacing, and positioning. CSS operates through a cascading mechanism, where styles can be inherited, overridden, or combined based on specificity and order of application. This separation of concerns enhances maintainability and flexibility in web development, as changes to the visual aspects of a website can be implemented globally by modifying the CSS, without altering the underlying HTML structure.

## **6.6 BACKEND:**

SQLite is a lightweight, serverless, and self-contained relational database management system. It is an ideal choice for small to medium-scale applications or prototypes due to its simplicity and minimal configuration requirements. For the Stroke Prediction System, SQLite can be used as the backend database to store user data and predictions securely.

## **7. SYSTEM REQUIREMENTS SPECIFICATIONS**

### **7.1 Software Requirements**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter
- HTML, CSS
- SQL
- VS code

### **7.2 Hardware Requirements**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- System : Pentium IV 2.4 GHz.
- Hard Disk : 100 GB.
- Monitor : 15 VGA Color.
- Mouse : Logitech.
- RAM : 1 GB.

## **8. FUNCTIONAL REQUIREMENTS**

### **8.1 Output Design**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

#### **Output Definition**

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

### **8.2 Input Design**

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

## **Input Stages**

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

## **Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

## **Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said

that most of the inputs are of the form of internal and interactive. As Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

### **Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

### **Error Detection**

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

### **Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

## **8.3 User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

### **User Interface Systems Can Be Broadly Classified As:**

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response.



## **User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

## **Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

## **Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

## **8.4 Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate

All the other requirements which do not form a part of the above specification are

categorized as Non-Functional needs. A system perhaps needed to gift the user with a show of the quantity of records during info. If the quantity must be updated in real time, the system architects should make sure that the system is capable of change the displayed record count at intervals associate tolerably short interval of the quantity of records dynamic. Comfortable network information measure may additionally be a non-functional requirement of a system.

The following are the features:

- Accessibility
- Availability
- Backup
- Certification
- Compliance
- Configuration Management
- Documentation
- Disaster Recovery
- Efficiency (resource consumption for given load)
- Interoperability

## **8.5 Feasibility Study**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

### **8.5.1 Technical Feasibility**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the

number or location of users?

- Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

### **8.5.2 Operational Feasibility**

#### **User-friendly**

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

#### **Reliability**

The package will pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

#### **Security**

The web server and database server should be protected from hacking, virus etc

#### **Portability**

The application will be developed using standard open source software (Except Oracle) like Java, tomcat web server, Internet Explorer Browser etc these software will work both on Windows and Linux o/s. Hence portability problems will not arise.

#### **Availability**

This software will be available always.

#### **Maintainability**

The system uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses My-Sql, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

### **8.5.3 Economic Feasibility**

The computerized system takes care of the present existing system's data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports. It should be built as a web based application with separate web server and database server. This is required as the activities are spread throughout the organization customer wants a centralized database. Further some of the linked transactions take place in different locations.

## 9.METHODOLOGY

## 9.1 Requirements Gathering Stage

## SDLC (Software Development Life Cycle) – Umbrella Model

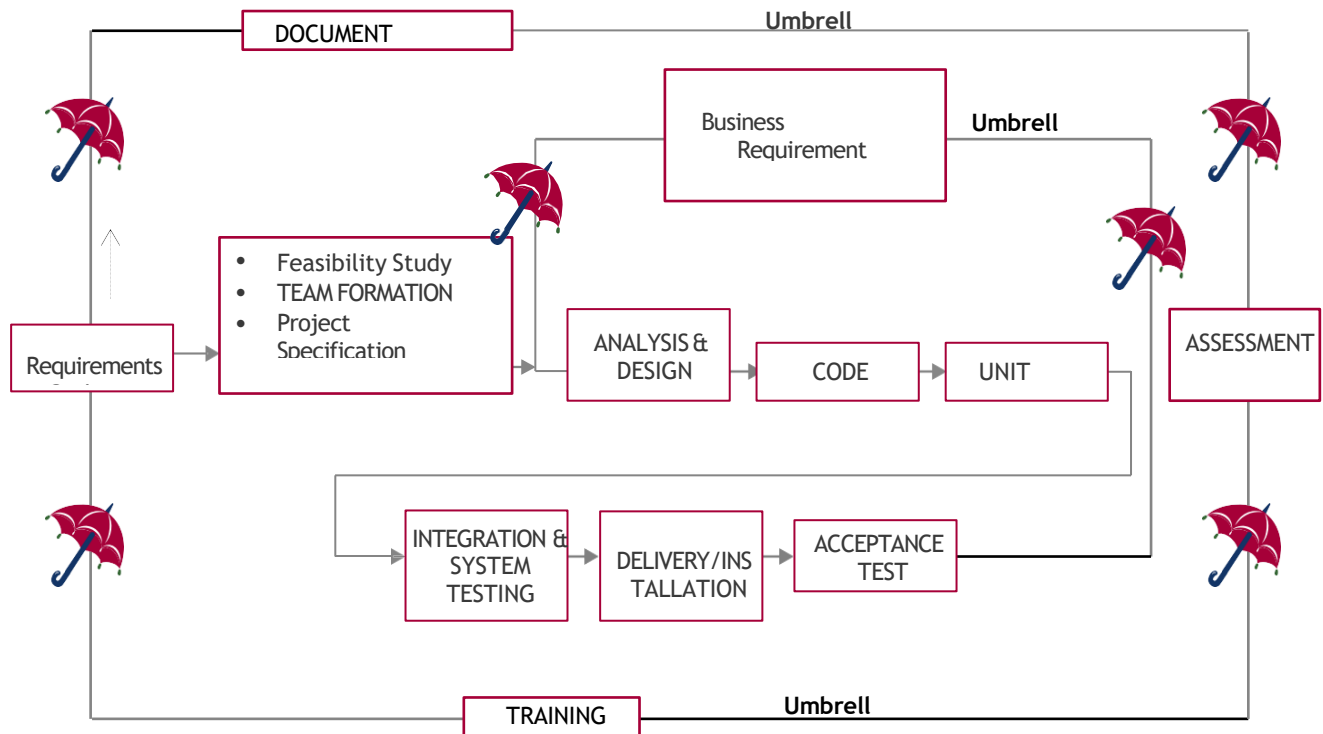


Fig no. 9.1 Umbrella model

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

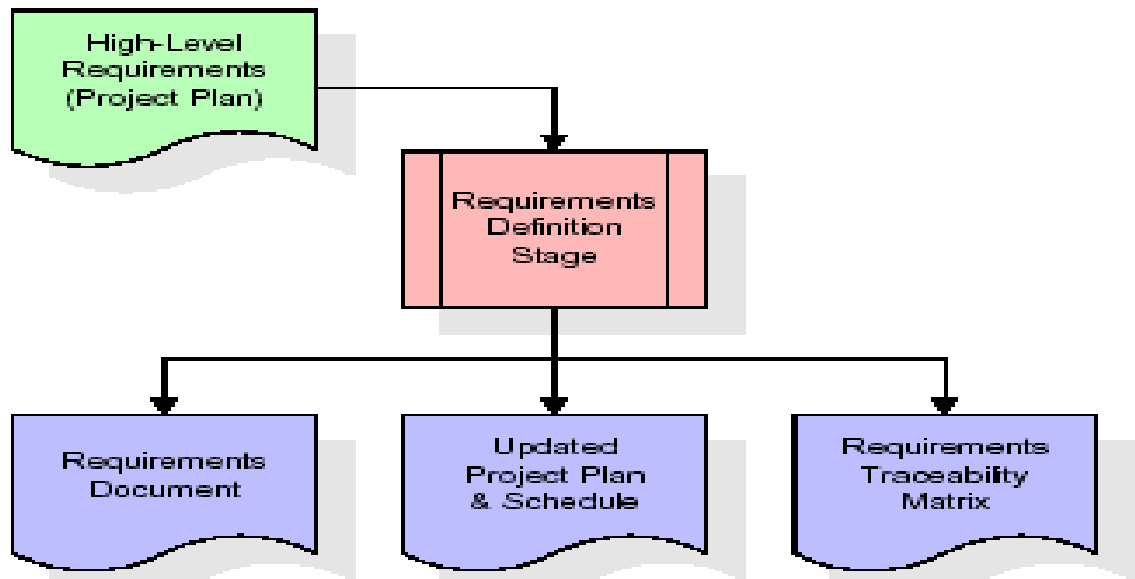


Fig no. 9.2 Requirements Gathering stage

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are not included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

Feasibility study is all about identification of problems in a project, number of staff required to handle a project is represented as Team Formation, in this case only modules

are individual tasks will be assigned to employees who are working for that project. Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

## 9.2 Analysis Stage

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.

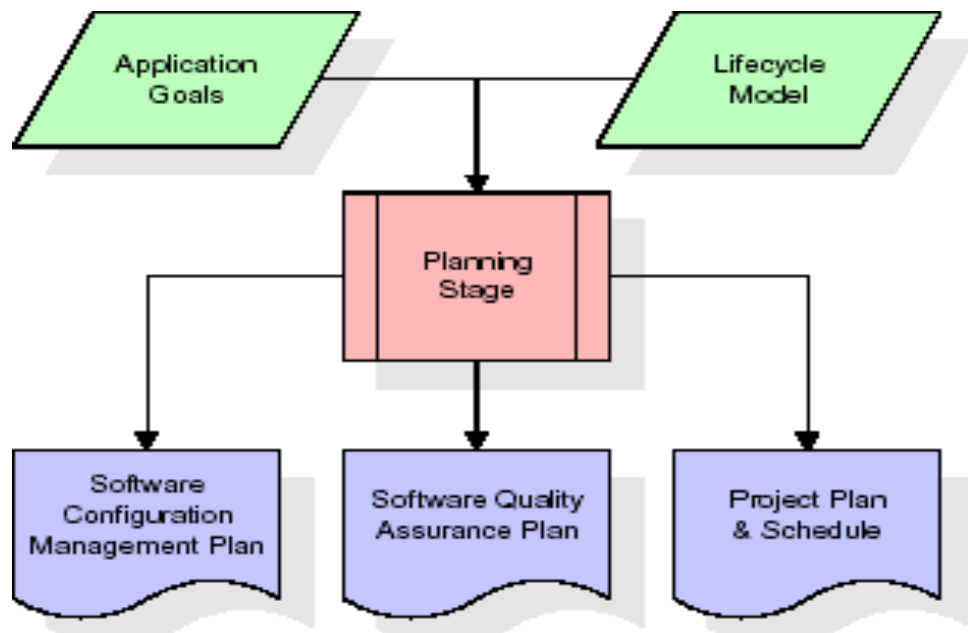


Fig no. 9.3 Analysis stage

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

### 9.3 Designing Stage

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

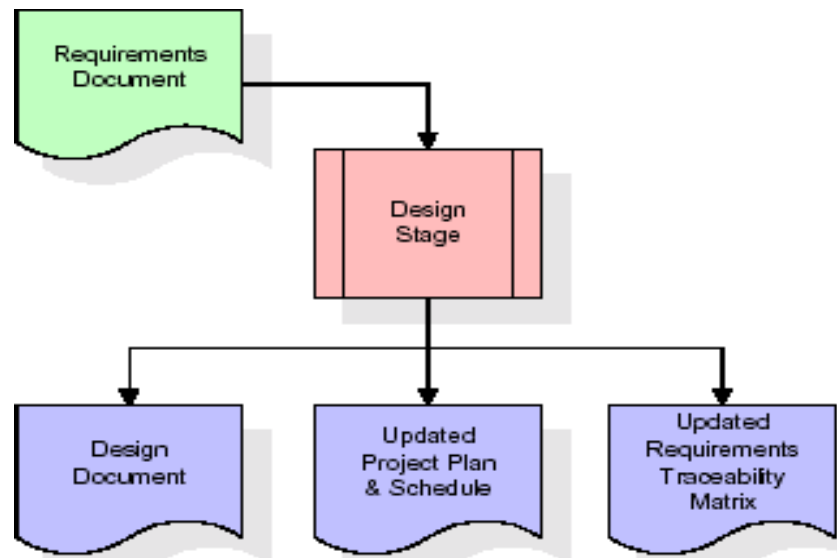


Fig no. 9.4 Designing stage

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

### 9.4 Development (Coding) Stage

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and

functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

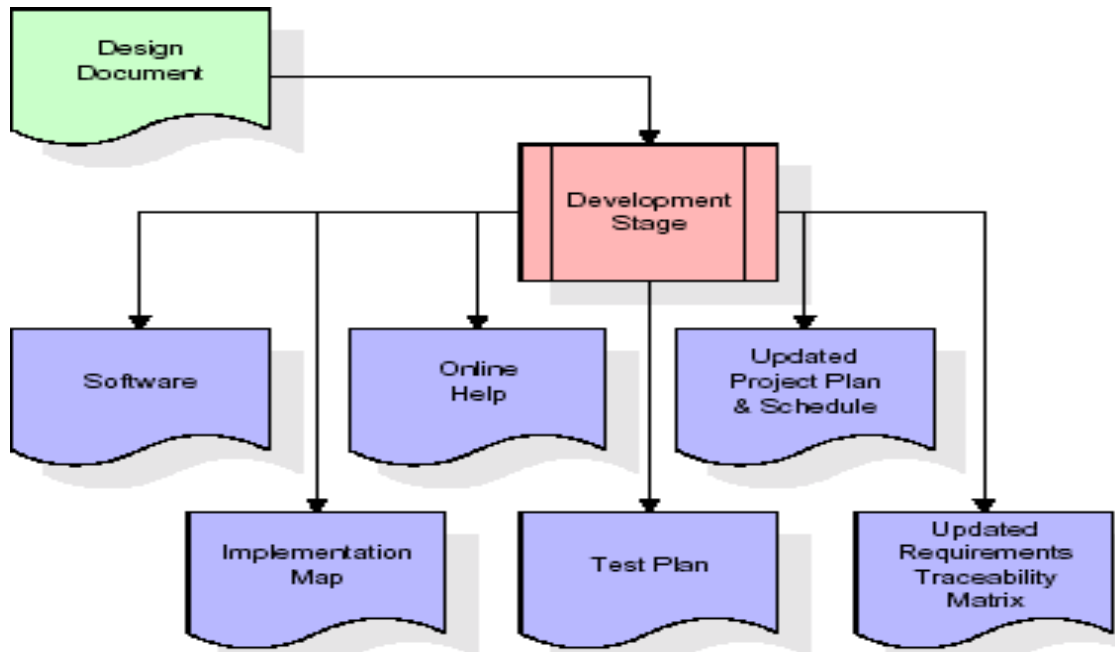


Fig no. 9.5 Coding stage

## 9.5 Integration & Test Stage

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite

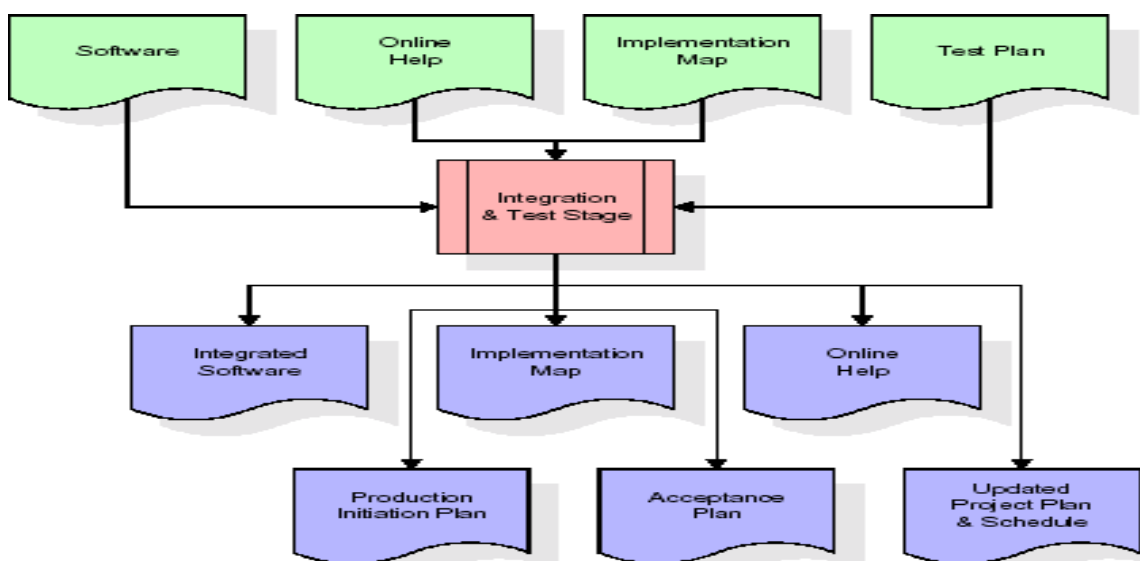


Fig no. 9.6 Integration and Testing Stage



confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

## 9.6 Installation & Acceptance Test

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.

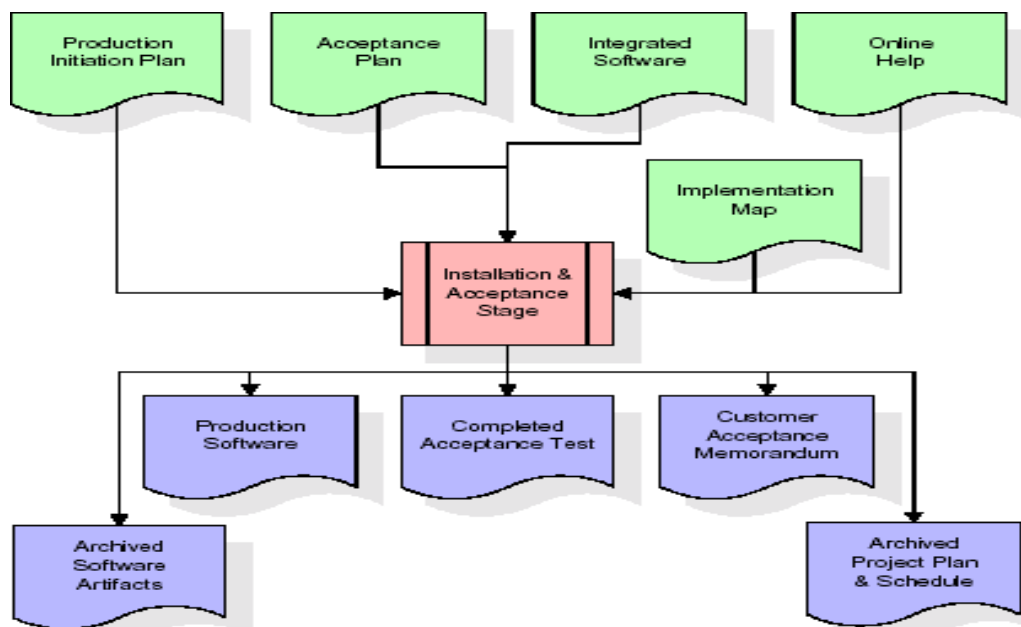


Fig no.9.7 Installation

## 9.7 Maintenance

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

## **10. SYSTEM TESTING**

### **10.1 TESTING**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. The following is the description of the testing strategies, which were carried out during the testing period.

### **10.2 SYSTEM TESTING**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

### **10.3 MODULE TESTING**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately.

In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

#### **10.4 INTEGRATION TESTING**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system

#### **10.5 ACCEPTANCE TESTING**

When that user find no major problems with its accuracy, the system passes through a final acceptance test. This test confirms that the system meets the original goals, objectives and requirements established during analysis without actual execution which eliminates wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

## 11.SOURCE CODE

```
@app.route("/")
def home():
    return render_template('home.html') # Render home page for all users

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

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = generate_password_hash(request.form['password'])
        add_user(username, password)
        return redirect(url_for('login')) # Redirect to login after registration
    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = get_user(username)
        if user and check_password_hash(user[2], password): # Validate credentials
            session['username'] = username # Store username in session
            return redirect(url_for('index')) # Redirect to home after successful login
        else:
            return render_template('login.html', error='Invalid credentials') # Pass error
message to the template
    return render_template('login.html') # Show login form

@app.route('/logout')
def logout():
```

```

session.pop('username', None) # Clear session
return redirect(url_for('home')) # Redirect to home page after logout

@app.route('/result', methods=['GET', 'POST'])
def predict():
    if 'username' not in session: # Check if user is logged in
        return redirect(url_for('login')) # Redirect to login if not

    if request.method == "POST":
        # Extract features from the form
        gender_Male = int(request.form['gender'])
        age = int(request.form['age'])
        hypertension_1 = int(request.form['hypertension'])
        heart_disease_1 = int(request.form['disease'])
        ever_married_Yes = int(request.form['married'])
        work = int(request.form['work'])
        Residence_type_Urban = int(request.form['residence'])
        avg_glucose_level = float(request.form['avg_glucose_level'])
        bmi = float(request.form['bmi'])
        smoking = int(request.form['smoking'])

        # Work type encoding
        work_type_Never_worked = 1 if work == 1 else 0
        work_type_Private = 1 if work == 2 else 0
        work_type_Self_employed = 1 if work == 3 else 0
        work_type_children = 1 if work == 4 else 0

        # Smoking status encoding
        smoking_status_formerly_smoked = 1 if smoking == 1 else 0
        smoking_status_never_smoked = 1 if smoking == 2 else 0
        smoking_status_smokes = 1 if smoking == 3 else 0

        # Prepare input features for prediction
        input_features = [age, avg_glucose_level, bmi, gender_Male, hypertension_1,
heart_disease_1,ever_married_Yes, work_type_Never_worked, work_type_Private,

```

```

work_type_Self_employed, work_type_children,
Residence_type_Urban, smoking_status_formerly_smoked,
smoking_status_never_smoked, smoking_status_smokes]

# Create DataFrame for prediction
df = pd.DataFrame([input_features], columns=['age', 'avg_glucose_level', 'bmi',
'gender_Male', 'hypertension_1', 'heart_disease_1', 'ever_married_Yes',
'work_type_Never_worked', 'work_type_Private', 'work_type_Self-employed',
'work_type_children', 'Residence_type_Urban', 'smoking_status_formerly
smoked', 'smoking_status_never smoked', 'smoking_status_smokes'])

# Make prediction
prediction = model.predict(df)[0]

# Save the user input and prediction to the database
username = session['username']

save_user_input(username, gender_Male, age, hypertension_1, heart_disease_1,
ever_married_Yes, Residence_type_Urban, avg_glucose_level, bmi, smoking,
prediction)

# Render prediction result
if prediction == 1:
    return render_template('index.html', prediction_text='Patient has stroke risk')
else:
    return render_template('index.html', prediction_text='Congratulations, patient
does not have stroke risk')

if __name__ == "__main__":
    app.run(debug=True) # Run the app in debug mode for better error messages
import sqlite3
def init_db():
    try:
        with sqlite3.connect('users.db') as conn:
            c = conn.cursor()
            # Create users table if not exists
            c.execute("""
                CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,

```

```

        username TEXT NOT NULL UNIQUE,
        password TEXT NOT NULL
    )
    """)
# Create user_inputs table to store prediction inputs
c.execute("""
INSERT INTO user_inputs (username, gender, age, h
ypertension,heart_disease, ever_married,work, residence, avg_glucose_level, bmi,
smoking, prediction)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

", (username, gender, age, hypertension, heart_disease, ever_married, work,
residence, avg_glucose_level,
    bmi, smoking, prediction))
conn.commit()
except sqlite3.Error as e:
    print(f"An error occurred while saving user input: {e}")

```

```

CREATE TABLE IF NOT EXISTS user_inputs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT,
    gender INTEGER,
    age INTEGER,
    hypertension INTEGER,
    heart_disease INTEGER,
    ever_married INTEGER,
    work INTEGER,
    residence INTEGER,
    avg_glucose_level REAL,
    bmi REAL,
    smoking INTEGER,
    prediction INTEGER,
    FOREIGN KEY (username) REFERENCES users(username)
)
""")
conn.commit()

```

```

except sqlite3.Error as e:
    print(f"An error occurred while initializing the database: {e}")

def add_user(username, password):
    try:
        with sqlite3.connect('users.db') as conn:
            c = conn.cursor()
            c.execute('INSERT INTO users (username, password) VALUES (?, ?)',
(username, password))
            conn.commit()
    except sqlite3.Error as e:
        print(f"An error occurred while adding a user: {e}")

def get_user(username):
    try:
        with sqlite3.connect('users.db') as conn:
            c = conn.cursor()
            c.execute('SELECT * FROM users WHERE username = ?', (username,))
            user = c.fetchone()
            return user
    except sqlite3.Error as e:
        print(f"An error occurred while fetching the user: {e}")
        return None

def save_user_input(username, gender, age, hypertension, heart_disease, ever_married,
work, residence,
                    avg_glucose_level, bmi, smoking, prediction):
    try:
        with sqlite3.connect('users.db') as conn:
            c = conn.cursor()
            c.execute("""
                INSERT INTO user_inputs (username, gender, age, hypertension,
heart_disease, ever_married, work, residence, avg_glucose_level, bmi, smoking,
prediction)
            # Import necessary libraries
import numpy as np
import pandas as pd

```



```

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Load the dataset
df = pd.read_csv("healthcare-dataset-stroke-data.csv")

# Check for missing values and fill or drop as appropriate
df.fillna(df.mean(), inplace=True)

# Encode categorical features
label_encoder = LabelEncoder()
df['gender'] = label_encoder.fit_transform(df['gender'])
df['ever_married'] = label_encoder.fit_transform(df['ever_married'])
df['work_type'] = label_encoder.fit_transform(df['work_type'])
df['Residence_type'] = label_encoder.fit_transform(df['Residence_type'])
df['smoking_status'] = label_encoder.fit_transform(df['smoking_status'])

# Split the dataset into features and target
X = df.drop('stroke', axis=1) # Features
y = df['stroke']             # Target variable

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

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#### Decision Tree Classifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
dt_pred = dt_model.predict(X_test)

print("Decision Tree Accuracy:", accuracy_score(y_test, dt_pred))
print(classification_report(y_test, dt_pred))

#### Naive Bayes Classifier

```

```

nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_pred = nb_model.predict(X_test)
print("Naive Bayes Accuracy:", accuracy_score(y_test, nb_pred))
print(classification_report(y_test, nb_pred))

### Artificial Neural Network (ANN)
ann_model = Sequential()
ann_model.add(Dense(32, activation='relu', input_shape=(X_train.shape[1],)))
ann_model.add(Dense(16, activation='relu'))
ann_model.add(Dense(1, activation='sigmoid'))

# Compile the model
ann_model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Train the model
ann_model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
# Evaluate the model
ann_pred = (ann_model.predict(X_test) > 0.5).astype("int32")
print("ANN Accuracy:", accuracy_score(y_test, ann_pred))
print(classification_report(y_test, ann_pred))

```

## 12. RESULTS AND DISCUSSIONS

### 12.1 Implementation Description

#### 12.1.1 Machine Learning Model Training and Evaluation

**Purpose:** This module is used for training and evaluating the machine learning models that predict stroke risk. Although it may not be part of the running system, this module is crucial for the initial system setup and improvement.

**Key Components:**

- **Model Training:** Trains machine learning models like ANN, Decision Tree, and Naive Bayes on historical health data.
- **Model Evaluation:** Evaluates the models' accuracy and performance using appropriate metrics (e.g., accuracy, precision, recall).
- **Model Saving:** Saves the trained models to disk using pickle, so they can be loaded and used by the prediction module.

#### 12.1.2 User Registration and Authentication

**Purpose:** This module allows users to register and log in to the system securely. It stores user credentials, hashes the passwords for security, and handles user authentication during login.

**Key Components:**

- **Registration Page:** Where new users can sign up by entering a username and password.
- **Login Page:** Allows users to log in using their credentials.
- **Password Hashing and Validation:** Ensures secure handling of user passwords using hashing techniques.
- **Session Management:** Manages user sessions, storing user information in session variables to maintain the user's logged-in status.

#### 12.1.3 Stroke Prediction Module

**Purpose:** This model uses machine learning Models to predict the risk

**Key Components:**

- **Model Loading:** Loads the pre-trained machine learning models from disk

(pickle files)

- **Data Transformation:** Converts the user input into a format that the model can process (like a pandas Data Frame).**Prediction Execution:** Runs the input data through the machine learning models to obtain a prediction (stroke risk or no stroke risk).
- **Output Display:** Displays the result of the prediction to the user, indicating whether the patient has a stroke risk.

## 12.2 Results

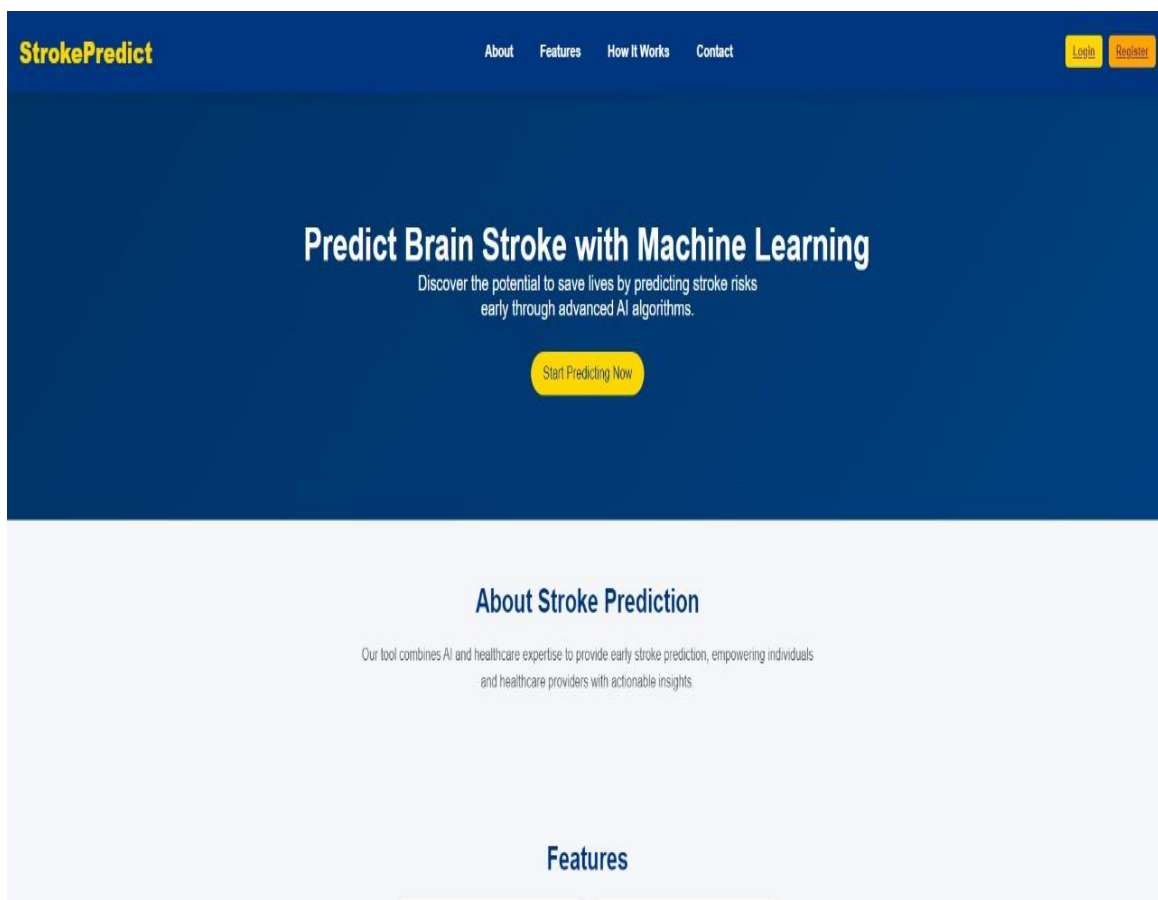


Fig:12.1 Home page

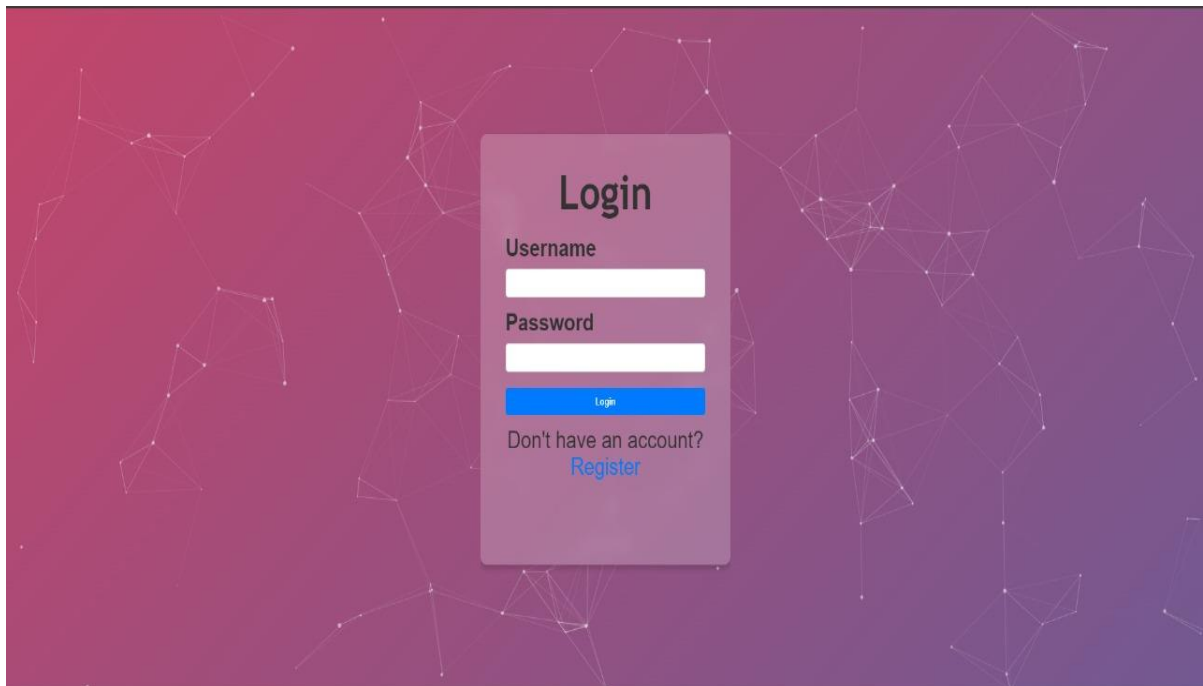


Fig.12.2 User Login Page

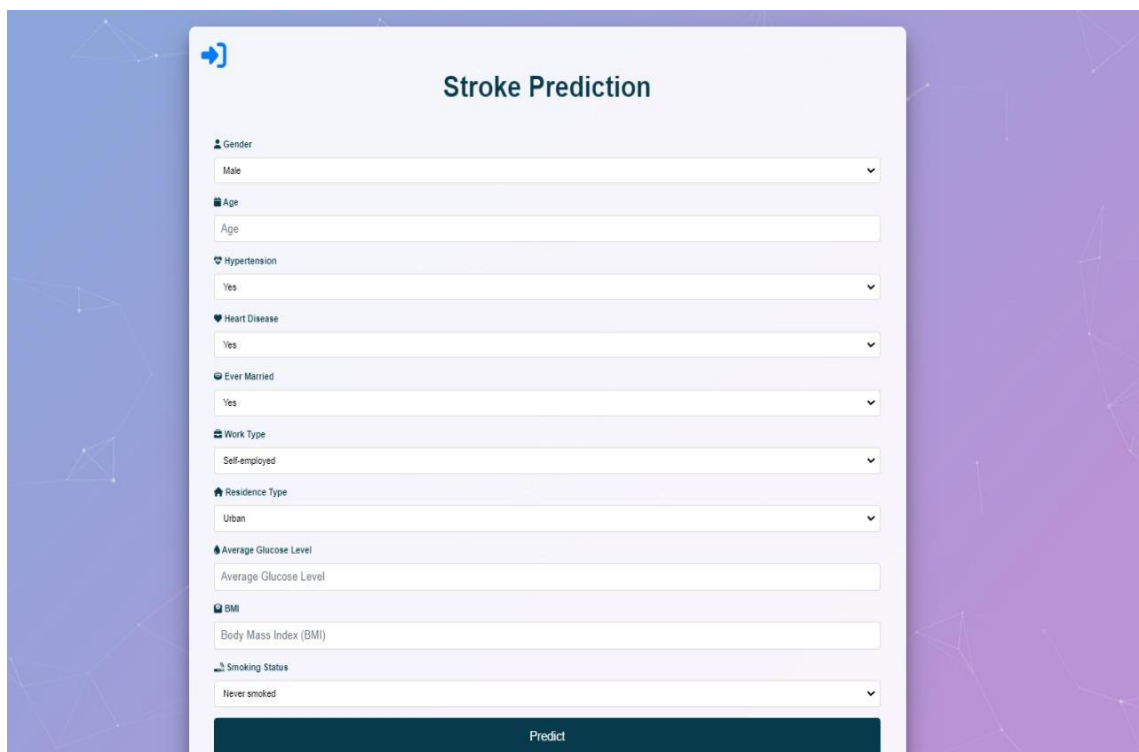
A "Stroke Prediction" form with a light purple header and a white body. The form includes several input fields with icons: Gender (person icon), Age (calendar icon), Hypertension (heart with pulse icon), Heart Disease (heart icon), Ever Married (wedding ring icon), Work Type (factory icon), Residence Type (house icon), Average Glucose Level (syringe icon), BMI (person with scale icon), and Smoking Status (cigarette icon). Each field has a dropdown arrow. A dark blue "Predict" button is at the bottom. The background is a purple gradient with a network pattern.

Fig 12.3 Prediction Page

### Stroke Prediction

Gender: Male

Age:

Hypertension: Yes

Heart Disease: Yes

Ever Married: Yes

Work Type: Self-employed

Residence Type: Urban

Average Glucose Level:

BMI: Body Mass Index (BMI)

Smoking Status: Never smoked

Predict

**Congratulations, patient does not have stroke risk**

Fig:12.4 Result Page

Yes

Work Type: Self-employed

Residence Type: Urban

Average Glucose Level:

BMI: Body Mass Index (BMI)

Smoking Status: Never smoked

Predict

**Patient has stroke risk**

SAMSUNG

Fig.12.5 Result Page

## 13. CONCLUSION AND FUTURE SCOPE

### 13.1 Conclusion

In this project, a comprehensive stroke prediction system has been developed using machine learning algorithms. The goal of the system is to predict the likelihood of a person having a stroke based on a set of health and lifestyle factors, which could serve as an early warning mechanism to help individuals take proactive measures in preventing stroke-related complications. The system integrates three different machine learning algorithms: Artificial Neural Networks (ANN), Decision Tree, and Naive Bayes — to predict the stroke risk with high accuracy. Each of these models was trained using relevant features, including age, BMI, blood glucose levels, smoking habits, and work type, among others. The ensemble approach of using multiple models enhances the reliability and robustness of the system by reducing the risk of misclassification and improving generalization across different types of data. The Artificial Neural Network model provided excellent results in detecting complex relationships within the dataset, demonstrating the power of deep learning techniques in medical prediction tasks. The Decision Tree model, with its transparent and interpretable structure, was able to classify the data efficiently by making decisions based on feature importance. Meanwhile, the **Naive Bayes** classifier, with its simplicity and speed, delivered reliable predictions and proved effective in dealing with large amounts of data with minimal computational overhead.

#### Contributions and Advantages

- It promotes awareness and provides valuable insights for early diagnosis, which can help individuals take preventive measures, seek medical attention, or follow lifestyle changes to mitigate the risk of stroke.
- The modularity of the system makes it flexible for future enhancements. Additional models, more features, or even a mobile application version could be integrated to extend its utility.

### 13.2 Future Scope

To enhance the precision of predictions, more sophisticated models, such as ensemble methods or hybrid machine learning approaches, could be explored. Real-time data integration through mobile apps or wearable devices could be incorporated to provide dynamic predictions based on ongoing health measurements. The system could be expanded to include more comprehensive health-related data (e.g., medical history, family history) to provide more detailed and personalized risk assessments.

## References

- [1] K. Yazhini and D. Loganathan, "A State of Art Approaches on Deep Learning Models in Healthcare: An Application Perspective", 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2020.
- [2] Suresh K. Peddoju, Himanshu Upadhyay and Shekhar Bhansali, "Health Monitoring with Low Power IoT Devices using Anomaly Detection Algorithm", Fourth International Conference on Fog and Mobile Edge Computing (FMEC), 2021.
- [3] Pattanapong Chantamit and Madhu Goyal, "Prediction of Stroke Using Deep Learning Model" in ICONIP, Springer International Publishing, pp. 774-781, 2020.
- [4] Heo JoonNyung and G. Yoon Jihoon, "Machine Learning-Based Model for Prediction of Outcomes in Acute Stroke", American Heart Association, vol. 50, pp. 00-00, 2020.
- [5] X. Zhang, X. Wei, F. Li, F. Hu, W. Jia and C. Wang, "Fuzzy Support Vector Machine with Imbalanced Regulator and its Application in Stroke Classification, 4–9 April 2022.
- [6] Songhee Cheon, Jungyoon Kim and Jihye Lim, The Use of Deep Learning to Predict Stroke Patient Mortality, Korea:Department of Physical Therapy, Youngsan University, Yangsan, pp. 626-790, April 2021.
- [7] Alharbi, W. Alosaimi, R. Sahal and H. Saleh, "Real-time system prediction for heart rate using deep learning and stream processing platforms", Complexity, vol. 2021, 2021.
- [8] L. Ding, C. Liu, Z. Li and Y. Wang, "Incorporating artificial intelligence into stroke care and research", Stroke, vol. 51, no. 12, pp. e351-e354, 2020.
- [9] S. Dev, H. Wang, C. S. Nwosu, N. Jain, B. Veeravalli and D. John, "A predictive analytics approach for stroke prediction using machine learning and neural networks", Healthcare Analytics, vol. 2, pp. 100032, 2022.
- [10] K. Kim, Y. J. Choo and M. C. Chang, "Prediction of motor function in stroke patients using machine learning algorithm: Development of practical models", Journal of Stroke and Cerebrovascular Diseases, vol. 30, no. 8, pp. 105856, 2021.