A

PROJECT REPORT ON

HEART DISEASE PREDICTION THROUGH MACHINE LEARNING IN DIGITAL HEALTHCARE

Submitted in partial fulfillment of the requirements

for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

By

MS. KANASANI MAMATHA,

(Regd.No:235N1F0058)

Under the Guidance of

K.VENKATA RAMYA,

Assistant Professor,

DEPARTMENT OF COMPUTER APPLICATIONS

ANNAMACHARYA P.G COLLEGE OF COMPUTER STUDIES

NEW BOYANAPALLI-516126, RAJAMPET (A.P)

(Approved by A.I.C.T.E., New Delhi & Affiliated to J.N.T.U.A,

Anantapuram, UGC(2f)Recognized Institution)

(2023-2025)

***

ANNAMACHARYA P.G COLLEGE OF COMPUTER STUDIES,

NEW BOYANAPALLI, RAJAMPET - 516126.

Affiliated to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

ANANTAPUR, ANANTAPURAMU

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project work entitled **"HEART DISEASE PREDICTION THROUGH MACHINE LEARNING IN DIGITAL HEALTHCARE"** is the Bonafide work carried out by **MS. KANASANI MAMATHA,** Regd. No: **235N1F0058** is submitted in the partial fulfillment of the requirements for the award of degree of **Master of Computer Applications** during the year,**2023-2025**

PROJECT GUIDE **PRINCIPAL**

External Examiner

DECLARATION

I, **KANASANI MAMATHA** hereby declare that the project report entitled as **"HEART DISEASE PREDICTION THROUGH MACHINE LEARNING IN DIGITAL HEALTHCARE",** is done original and independent record of work, submitted by me to **JNTUA**, Anantapuramu, under the guidance of **K.VENKATA RAMYA**, Assistant professor of Annamacharya P.G College of Computer Studies, Rajampet, for the award of the degree of Master of Computer Applications and has not been submitted either in part or in full for the award of any Degree or Diploma.

.

Place : Rajampet **KANASANI MAMATHA**

Date : **(Regd. No: 235N1F0058)**

ACKNOWLEDGEMENT

An endeavor over a long period can be successful only with the advice of many wellwishers. I take this opportunity to express my deep gratitude and appreciation of all those who encourage me to successfully complete the project.

I wish to express my sincere gratitude to **Dr. D.J.SAMATHA NAIDU**, Principal of **Annamacharya P.G College of Computer Studies, New Boyanapalli, Rajampet**, for her consistent help and providing such facilities to complete this project.

I express my sincere thanks to my guide **Mrs. K. VENKATA RAMYA** for her valuable guidance and suggestions in analyzing and testing throughout the period of my project work.

Last but not least, I would like to thank my friends, teaching and non- teaching, one and all those who helped me to complete this project successfully.

K.MAMATHA

(Regd. No : 235N1F0058)

ABSTRACT

ABSTRACT

Heart disease is one of the complex diseases and globally many people suffered from this disease. On time and efficient identification of heart disease plays a key role in healthcare, particularly in the field of cardiology. The proposed an efficient and accurate system to diagnosis heart disease and the system is based on machine learning techniques. The system is developed based on classification algorithms includes Support vector machine, Logistic regression, Artificial neural network, K-nearest neighbor, Naïve bays, and Decision tree while standard features selection algorithms have been used such as Relief, Minimal redundancy maximal relevance, Least absolute shrinkage selection operator and Local learning for removing irrelevant and redundant features and also proposed novel fast conditional mutual information feature selection algorithm to solve feature selection problem. Heart disease is one of the complex diseases and globally many people suffered from this disease. On time and efficient identification of heart disease plays a key role in healthcare, particularly in the field of cardiology and also proposed novel fast conditional mutual information feature selection algorithm to solve feature selection problem. The features selection algorithms are used for features selection to increase the classification accuracy and reduce the execution time of classification system. The experimental results show that the proposed feature selection algorithm is feasible with classifier support vector machine for designing a high-level intelligent system to identify heart disease.

CONTENTS

7 BIBLIOGRAPHY

Appendix-A

- URL Listing
- References

Appendix-B

- Glossary

Appendix-C

- List of Figures
- List of Tables
- List of Screens and Reports

Appendix-D

- Coding

INTRODUCTION

CHAPTER -1
INTRODUCTION

Heart disease (HD) is the critical health issue and numerous people have been suffered by this disease around the world. The HD occurs with common symptoms of breath shortness, physical body weakness and, feet are swollen . Researchers try to come across an efficient technique for the detection of heart disease, as the current diagnosis techniques of heart disease are not much effective in early time identification due to several reasons, such as accuracy and execution time . The diagnosis and treatment of heart and proper treatment can save the lives of many people . According to the European Society of Cardiology, 26 million approximately people of HD were diagnosed and diagnosed 3.6 million annually . Most of the people in the United States are suffering from heart disease . Diagnosis of HD is traditionally done by the analysis of the medical history of the patient, physical examination report and analysis of concerned symptoms by a physician. But the results obtained from this diagnosis method are not accurate in identifying the patient of HD. Moreover, it is expensive and computationally difficult to analyse. Thus, to develop a non in vasive diagnosis system based on classifiers of machine learning (ML) to resolve these issues. Expert decision system based on machine learning classifiers and the application of artificial fuzzy logic is effectively diagnosis the HD as a result, the ratio of death decreases and . The Clevel and heart disease data set was used by various researchers and for the identification problem of HD. The machine learning predictive models need proper data for training and testing. The performance of machine learning model can be increased if balanced dataset is use for training and testing of the model. Furthermore, the model predictive capabilities can improved by using proper and related features from the data. Therefore, data balancing and feature selection is significantly important for model performance improvement. In literature various diagnosis techniques have been proposed by various researchers, however these techniques are not effectively

diagnosis HD. In order to improve the predictive capability of machine learning model data preprocessing is important for data standardization. Various Preprocessing techniques such removal of missing feature value instances from the dataset, Standard Scalar (SS), Min-Max Scalar etc. The feature extraction and selection techniques are also improve model performance. Various feature selection techniques are mostly used for important feature selection such as, Least-absolute-shrinkage-selection-operator (LASSO), Minimal-Redundancy-Maximal-Relevance(MRMR), Local-learning-based-features-selection (LLBFS), Principle component Analysis (PCA), Greedy Algorithm (GA), and optimization methods, such as Anty Conley Optimization (ACO), fruit fly optimization (FFO), Bacterial Foraging Optimization (BFO) etc. Similarly Yun *et al.* presented different techniques for different type of feature selection, such as feature selection for high-dimensional small sample size data, large-scale data, and secure feature selection. They also discussed some important topics for feature selection have emerged, such as stable feature selection, multi-view feature selection, distributed feature selection, multi-label feature selection, online feature selection, and adversarial feature selection. Jundong *et al.* discussed the challenges of feature selection (FS) for big data. It is necessary to decrease the dimensionality of data for various learning tasks due to the curse of dimensionality. Feature selection has great influence in numerous applications such as building simpler, increasing learning performance, creating clean and understandable data. The feature selection from big data is challenging job and create big problems because big data has many dimensions. Further, challenges of feature selection for structured, heterogeneous and streaming data as well as its scalability and stability issues. For big data analytics challenges of feature selection is very important to resolved. In designed unsupervised hashing scheme, called topic hyper graph hashing, to report the limitations. Topic hypergraph hashing effectively mitigates the semantic shortage of hashing codes by exploiting auxiliary text around images. The proposed Topic hyper graph hashing can achieve superior performance equal with numerous state-of-theart approaches, and it is more appropriate for mobile image retrieval. The feature selection algorithms are classified into three type such as filter

based, wrapper based and embedded based. All these feature selection mechanisms have some advantages and limitations in certain cases. The filter based method measures the relevance of a feature by correlation with the dependent variable while the wrapper feature selection algorithm measure the usefulness of a subset of features by actually training the classifier on it. The filter method is less computationally complex than wrapper method. The feature set selected by the filter is general and can be applied to any model and it is independent of a specific model.

In feature selection global relevance is of greater importance.

On another hand suitable machine learning model is necessary for good results. Obviously, a good machine learning model is a model that not only performs well on data seen during training (else a machine learning model could simply learn the training data), but also on unseen data. To evaluate all classifiers on data and find that they get, on average, 50% of the cases right [16]. Furthermore, appropriate cross validation techniques and performance evaluation metrics are critical necessary for a model when model is train and test on dataset.

The proposed a machine learning based diagnosis method for the identification of HD in this research work. Machine learning predictive models include ANN, LR, K-NN, SVM, DT, and NB are used for the identification of HD. The standard state of the art features selection algorithms, such as Relief, mRMR, LASSO and Local-learning-based features-selection (LLBFS) have been used to select the features and also proposed fast conditional mutual information (FCMIM) features selection algorithm for features selection. Leave-one-subject-out cross-validation (LOSO) technique has been applied to select the best hyper-parameters for best model selection. Apart from this, different performance assessment metrics have been used for classifiers performances evaluation. The proposed method has been tested on Cleveland HD dataset. Furthermore, the performance of the proposed technique have been compared with state of the art existing methods in the literature, such as NB , Three phase ANN (Artificial neural Network) diagnosis system , Neural network ensembles

(NNE) , ANN-Fuzzy-AHP diagnosis system (AFP), Adaptive-weighted-Fuzzy-system-ensemble (AWFSE) . The research study has the following contributions.

Firstly, the authors try to address the problem of features selection by employing pre-processing techniques and standard state of the art four features selection algorithms such as Relief, mRMR, LASSO, and LLBFS for appropriate subset of features and then applied these features for effective training and testing of the classifiers that identify which feature selection algorithm and classifier gives good results in term of accuracy and computation time.

Secondly, the authors proposed fast conditional mutual information (FCMIM) FS algorithm for feature selection and then these features are input to classifiers for improving prediction accuracy and reducing computation time. The classifiers performances have been compared on features selected by the standard state of the art FS algorithms with the selected features of the proposed FS algorithm.

Thirdly, identify weak features from the dataset which affect the performance of the classifiers.

Finally, suggests that heart disease identification system (FCMIM-SVM) effectively identify the HD.

The paper remaining sections are structured as follows. The literature related to the problem has been discussed in section 2. In section 3 the dataset and the theoretical and mathematical knowledge of feature selection and classification algorithms are discussed in details. Additionally, discuss the technique of cross-validation and performance measuring metrics. In section 4 results of all experiments are analysed and discussed in details. The last section 5 the conclusion and future direction of the research work have been explored in details.

Disadvantages

Data Quality Issues: Machine learning models heavily depend on the quality and quantity of data. Inadequate or biased data can lead to inaccurate predictions. In healthcare, data may be incomplete, inconsistent, or biased, which can affect the performance of machine learning algorithms.

Interpretability: Some machine learning models, particularly complex ones like deep neural networks, lack interpretability. This means it can be challenging to understand how the model arrives at its predictions, which may limit trust from healthcare professionals and patients.

Overfitting and Generalization: Machine learning models may overfit to the training data, capturing noise or specific patterns that do not generalize well to unseen data. This can result in poor performance when applied to real-world patient data.

Privacy Concerns: Healthcare data is sensitive and subject to strict privacy regulations. Sharing patient data for training machine learning models raises concerns about patient privacy and data security. Unauthorized access or breaches could lead to serious consequences for patients and healthcare providers.

Bias and Fairness: Machine learning algorithms can inherit biases present in the training data, leading to unfair or discriminatory outcomes. Biases in healthcare data, such as disparities in access to healthcare services, can perpetuate inequalities in the detection and treatment of heart disease.

Advantages

Early Detection: Machine learning classification enables the early detection of heart disease by analysing patient data and identifying individuals at risk of developing cardiovascular conditions.

Personalized Risk Assessment: The proposed system can provide personalized risk assessments based on individual patient characteristics, allowing for targeted interventions and treatment plans.

Efficiency: Machine learning algorithms can automate the heart disease detection process, leading to faster and more efficient diagnosis and treatment decisions.

Improved Accuracy: By leveraging large amounts of patient data, machine learning models can achieve higher accuracy in detecting heart disease compared to traditional diagnostic methods.

Cost-Effectiveness: Implementing machine learning-based heart disease detection in e-healthcare can potentially reduce healthcare costs by optimizing resource allocation and improving patient outcomes through early intervention and preventive care.

Data Preprocessing: Implementing rigorous data preprocessing techniques to address data quality issues, handle missing values, and mitigate biases in the training data.

Model Interpretability: Using interpretable machine learning models or techniques, such as decision trees or rule-based systems, to enhance the transparency and explain ability of the heart disease detection process.

Regularization Techniques: Applying regularization techniques, such as dropout or L1/L2 regularization, to prevent overfitting and improve the generalization performance of the machine learning models.

Privacy-Preserving Methods: Employing privacy-preserving techniques, such as federated learning or differential privacy, to protect patient privacy while training machine learning models on distributed healthcare data.

Bias Mitigation Strategies: Implementing bias detection and mitigation strategies to address biases in the training data and ensure fairness in the heart disease detection process

1.1PURPOSE

The purpose of this study is to develop a machine learning-based classification system for the early detection of heart disease in e-healthcare settings. By leveraging advanced algorithms and health data collected from electronic health records (EHRs), wearable devices, and diagnostic tests, the system aims to accurately identify individuals at risk of heart disease. This early detection enables timely interventions, personalized treatment plans, and improved patient outcomes.

1.2 SCOPE

The scope of this project encompasses the following areas:

Data Collection: Gathering diverse health data sources including EHRs, vital signs, medical imaging, lifestyle factors, and genetic information if available.

Feature Engineering: Extracting relevant features from the collected data to characterize cardiovascular health indicators such as blood pressure, cholesterol levels, electrocardiogram (ECG) signals, and physical activity patterns.

Machine Learning Model Development: Designing and training classification algorithms (e.g., logistic regression, decision trees, support vector machines) using labeled datasets to predict the likelihood of heart disease based on the extracted features.

System Integration: Implementing the developed model into existing e-healthcare platforms or creating standalone applications for seamless integration into clinical workflows.

Performance Evaluation: Assessing the accuracy, sensitivity, specificity, and clinical utility of the classification system through validation studies, comparative analysis with existing diagnostic methods, and feedback from healthcare professionals.

1.3. NEED FOR SYSTEM

There are several compelling reasons for implementing a machine learning-based heart disease detection system in e-healthcare:

Early Diagnosis: Heart disease is a leading cause of morbidity and mortality worldwide, and early detection is crucial for timely intervention and prevention of adverse cardiovascular events.

Personalized Medicine: Machine learning algorithms can analyze vast amounts of patient data to tailor treatment plans and interventions based on individual risk profiles, genetic predispositions, and co morbidities.

Remote Monitoring: E-healthcare platforms equipped with heart disease detection capabilities enable remote monitoring of patients' cardiovascular health, allowing for proactive intervention and reducing the need for frequent clinic visits.

Healthcare Resource Optimization: By automating the screening and triage process for heart disease, the system can help prioritize high-risk patients for further diagnostic evaluation and specialist consultation, optimizing healthcare resource allocation.

Patient Empowerment: Providing patients with access to personalized risk assessments and actionable insights empowers them to actively engage in their cardiovascular health management and adopt preventive measures.

By addressing these needs, the proposed system has the potential to revolutionize cardiovascular care delivery, improve patient outcomes, and alleviate the burden on healthcare systems. Additionally, it lays the foundation for advancing precision medicine initiatives and promoting proactive health management in the digital era.

SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATION

CHAPTER-2

SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATION

2.1 RELATED WORK

[1] A. L. Bui, T. B. Horwich, and G. C. Fonarow, ``Epidemiology and risk prole of heart failure,'' Nature Rev. Cardiol., vol. 8, no. 1, p. 30, 2011.

Heart failure (HF) is a major public health issue, with a prevalence of over 5.8 million in the USA, and over 23 million worldwide, and rising. The lifetime risk of developing HF is one in five. Although promising evidence shows that the age-adjusted incidence of HF may have plateaued, HF still carries substantial morbidity and mortality, with 5-year mortality that rival those of many cancers. HF represents a considerable burden to the health-care system, responsible for costs of more than $39 billion annually in the USA alone, and high rates of hospitalizations, readmissions, and outpatient visits. HF is not a single entity, but a clinical syndrome that may have different characteristics depending on age, sex, race or ethnicity, left ventricular ejection fraction (LVEF) status, and HF etiology. Furthermore, pathophysiological differences are observed among patients diagnosed with HF and reduced LVEF compared with HF and preserved LVEF, which are beginning to be better appreciated in epidemiological studies. A number of risk factors, such as ischemic heart disease, hypertension, smoking, obesity, and diabetes, among others, have been identified that both predict the incidence of HF as well as its severity. In this Review, we discuss key features of the epidemiology and risk profile of HF.

[2] M. Durairaj and N. Ramasamy, ``A comparison of the perceptive approaches for preprocessing the data set for predicting fertility success rate,'' Int. J. Control Theory Appl., vol. 9, no. 27, pp. 255260, 2016.

Medical diagnostics systems are evaluated by employing large information databases, but it endures many failures to extract data from Database. There is no sufficient tool available to discover the major relationship between the data. In such case, the core knowledge of healthcare data is extracted by applying the data mining methods. The extracted knowledge can be used for the perfect diagnosis and further treatment. Infertility is an emotional cause of fertility in all over the world over past years. Treatment for infertility includes set of procedures like IUI, IVF, ICSI, and GIFT to cure the disease. Predicting the success rate for the infertility treatment can be done by using the preprocessed data from the database. This paper explains the existing Preprocessing method and analyzes the accuracy of prediction rate after preprocessing. It is evident that the accuracy is increased up to 90% after preprocessing the raw data using the existing techniques. Hybriding different techniques together will provide a better result which is taken as the future direction of this work.

[3] L. A. Allen, L.W. Stevenson, K. L. Grady, N. E. Goldstein, D. D. Matlock, R. M. Arnold, N. R. Cook, G. M. Felker, G. S. Francis, P. J. Hauptman, E. P. Havranek, H. M. Krumholz, D. Mancini, B. Riegel, and J. A. Spertus, ``Decision making in advanced heart failure: A scientic statement from the American heart association,'' Circulation, vol. 125, no. 15, pp. 19281952, 2012.

Shared decision making for advanced heart failure has become both more challenging and more crucial as duration of disease and treatment options have increased. High-quality decisions are chosen from medically reasonable options and are aligned with values, goals, and preferences of an informed patient. The top 10 things to know about decision making in advanced heart failure care are listed in Table 1.

[4] S. Ghwanmeh, A. Mohammad, and A. Al-Ibrahim, ``Innovative articial neural networks-based decision support system for heart diseases diagnosis,'' J. Intell. Learn. Syst. Appl., vol. 5, no. 3, 2013, Art. no. 35396.

Heart diagnosis is not always possible at every medical center, especially in the rural areas where less support and care, due to lack of advanced heart diagnosis equipment. Also, physician intuition and experience are not always sufficient to achieve high quality medical procedures results. Therefore, medical errors and undesirable results are reasons for a need for unconventional computer-based diagnosis systems, which in turns reduce medical fatal errors, increasing the patient safety and save lives. The proposed solution, which is based on an Artificial Neural Networks (ANNs), provides a decision support system to identify three main heart diseases: mitral stenosis, aortic stenosis and ventricular septal defect. Furthermore, the system deals with an encouraging opportunity to develop an operational screening and testing device for heart disease diagnosis and can deliver great assistance for clinicians to make advanced heart diagnosis. Using real medical data, series of experiments have been conducted to examine the performance and accuracy of the proposed solution. Compared results revealed that the system performance and accuracy are acceptable, with a heart diseases classification accuracy of 92%.

2.1.1 Literature Survey

In literature various machine learning based diagnosis techniques have been proposed by researchers to diagnosis HD. This research study present some existing machine learning based diagnosis techniques in order to explain the important of the proposed work. Detrano *et al.* developed HD classification system by using machine learning classification techniques and the performance of the system was 77% in terms of accuracy. Cleveland dataset was utilized with the method of global evolutionary and with features selection method. In another study Gudadhe *et al.* developed a diagnosis system using multi-layer Perceptron and support vector machine (SVM) algorithms for HD classification and achieved accuracy 80.41%. Humar *et al.* designed HD classification system by utilizing a neural network with the integration of Fuzzy logic. The classification system achieved 87.4% accuracy. Resul *et al.* developed an ANN ensemble based diagnosis system for HD along with statistical measuring system enterprise miner (5.2) and obtained the accuracy of 89.01%, sensitivity 80.09%, and specificity

95.91%. Akil *et al.* designed a ML based HD diagnosis system. ANN-DBP algorithm along with FS algorithm and performance was good. Palaniappan *et al.* proposed an expert medical diagnosis system for HD identification. In development of the system the predictive model of machine learning, such as navies bays (NB), Decision Tree (DT), and Artificial Neural Network were used. The 86.12% accuracy was achieved by NB, ANN accuracy 88.12% and DT classifier achieved 80.4% accuracy. Olaniyi *et al.* developed a three-phase technique based on the artificial neural network technique for HD prediction in angina and achieved 88.89% accuracy. Samuel *et al.* developed an integrated medical decision support system based on artificial neural network and Fuzzy AHP for diagnosis of HD. The performance of the proposed method in terms of accuracy was 91.10%. Liu *et al.* proposed a HD classification system using relief and rough set techniques. The proposed method achieved 92.32% classification accuracy. In proposed a HD identification method using feature selection and classification algorithms. Sequential Backward Selection Algorithm (SBS FS) for Features Selection. The classifier K-Nearest Neighbor (K-NN) performance has been checked on full and on selected features set. The proposed method obtained high accuracy. In another study MOHAN *et al.* designed a HD prediction method by using hybrid machine learning techniques. He also proposed a new method for significant feature selection from the data for effective training and testing of machine learning classifier. They have been recorded 88.07% classification accuracy. Geweid *et al.* designed HD identification techniques by using improved SVM based duality optimization technique. In the above literature the proposed HD diagnosis methods limitation and advantages have been summarized in Table 1 for better understanding the important of our proposed approach. All these existing techniques used numerous methods to identify the HD at early stages. However, all these techniques have lack of prediction accuracy and high computation time for prediction of HD. According to Table 1 the prediction accuracy of HD detection method need further improvement for efficient and accurate detection at early stages for better treatment and recovery. Thus, the major issues in these previous approaches are low accuracy and high computation time and these might be due the use of

irrelevant features in dataset. In order to tackle these problems new methods are needed to detect HD correctly. The improvement in prediction accuracy is a big challenge and research gap.

2.1.2 Existing Algorithms

Artificial neural networks are finding many uses in the medical diagnosis application. The goal of this paper is to evaluate artificial neural network in disease diagnosis. Two cases are studied. The first one is acute nephritis disease; data is the disease symptoms. The second is the heart disease; data is on cardiac Single Proton Emission Computed Tomography (SPECT) images. Each patient classified into two categories: infected and non-infected. Classification is an important tool in medical diagnosis decision support. Feed-forward back propagation neural network is used as a classifier to distinguish between infected or non-infected person in both cases. The results of applying the artificial neural networks methodology to acute nephritis diagnosis based upon selected symptoms show abilities of the network to learn the patterns corresponding to symptoms of the person. In this study, the data were obtained from UCI machine learning repository in order to diagnosed diseases. The data is separated into inputs and targets. The targets for the neural network will be identified with 1's as infected and will be identified with 0's as non-infected. In the diagnosis of acute nephritis disease; the percent correctly classified in the simulation sample by the feed-forward back propagation network is 99 percent while in the diagnosis of heart disease; the percent correctly classified in the simulation sample by the feed-forward back propagation network is 95 percent.

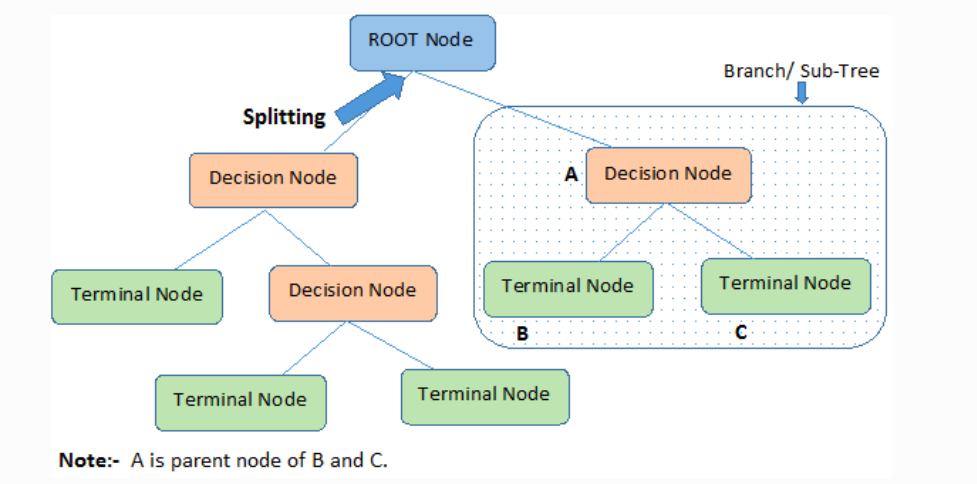
Decision tree classifiers

Decision tree classifiers are widely applied across various domains due to their ability to capture descriptive decision-making knowledge from provided data. One of their most notable features is their capability to be generated from training sets. The process for generating decision trees based on a set of objects (S), each belonging to one of the classes C1, C2, …, Ck, unfolds as follows:

Step 1: If all objects in S belong to the same class (e.g., Ci), the decision tree for S comprises a leaf label with this class.

Step 2: Otherwise, select a test (T) with potential outcomes (O1, O2,…, On). Each object in S has a specific outcome for test T, thus partitioning S into subsets (S1, S2,…, Sn), where each object in Si corresponds to outcome Oi for test T. Test T becomes the root of the decision tree, and for each outcome Oi, a subsidiary decision tree is constructed by recursively invoking the same procedure on set Si.

Block Diagram for Decision Tree Algorithm:



Root Node: It represents the entire population or sample and this further gets divided into two or more homogeneous sets.

Splitting: It is a process of dividing a node into two or more sub-nodes.

Decision Node: When a sub-node splits into further sub-nodes, then it is called the decision node.

Leaf / Terminal Node: Nodes do not split is called Leaf or Terminal node.

Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.

Branch / Sub-Tree: A subsection of the entire tree is called branch or sub-tree.

Gradient boosting

Gradient boosting is a versatile machine learning technique employed in regression and classification tasks, among others. It constructs a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When using decision trees as the weak learner, the resulting algorithm is referred to as gradient-boosted trees, often surpassing the performance of random forests. The construction of a gradient-boosted trees model occurs in a stage-wise manner, similar to other boosting methods, but it stands out by allowing the optimization of an arbitrary differentiable loss function.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a straightforward yet highly effective classification algorithm that operates based on a similarity measure. It is non-parametric and employs lazy learning, meaning it does not "learn" until presented with a test example. Whenever there is a new data point to classify, KNN identifies its K-nearest neighbours from the training data and determines its classification based on their majority vote or weighted vote.

Logistic regression

Logistic regression analysis explores the relationship between a categorical dependent variable and a set of independent variables. The term "logistic regression" is applied when the dependent variable has only two values, such as 0 and 1, or Yes and No. On the other hand, "multinomial logistic regression" is used when the dependent variable has three or more unique values, like Married, Single, Divorced, or Widowed. While the nature of data for the dependent variable differs from that of multiple regressions, the practical application of the procedure remains similar.

Logistic regression serves as a competitor to discriminant analysis in analysing categorical-response variables. Many statisticians logistic regression due to its versatility and suitability for model various situations compared to discriminant analysis. This preference arises from logistic regression's ability to not assume that the independent variables follow a normal distribution, unlike discriminant analysis.

Naive Bayes

The naive Bayes approach is a supervised learning method founded on a simple assumption: it presumes that the presence or absence of one feature of a class is independent of the presence or absence of any other feature. Despite its simplicity, it demonstrates robustness and efficiency comparable to other supervised learning techniques. One explanation often highlighted in the literature is based on representation bias.

The naive Bayes classifier operates as a linear classifier, akin to linear discriminant analysis, logistic regression, or linear support vector machines (SVMs). However, the distinction lies in the method used to estimate the classifier's parameters, known as the learning bias. Although the naive Bayes classifier finds extensive use in the research community due to its ease of programming, parameter estimation simplicity, rapid learning even with large datasets, and reasonably good accuracy compared to other methods, it remains less popular among practitioners seeking practical results. Researchers appreciate its simplicity and efficacy. However, practitioners often struggle with its interpretability and deployment, as they may not grasp its relevance or utility.

2.2 Research Methodology

2.2.1. System Architecture

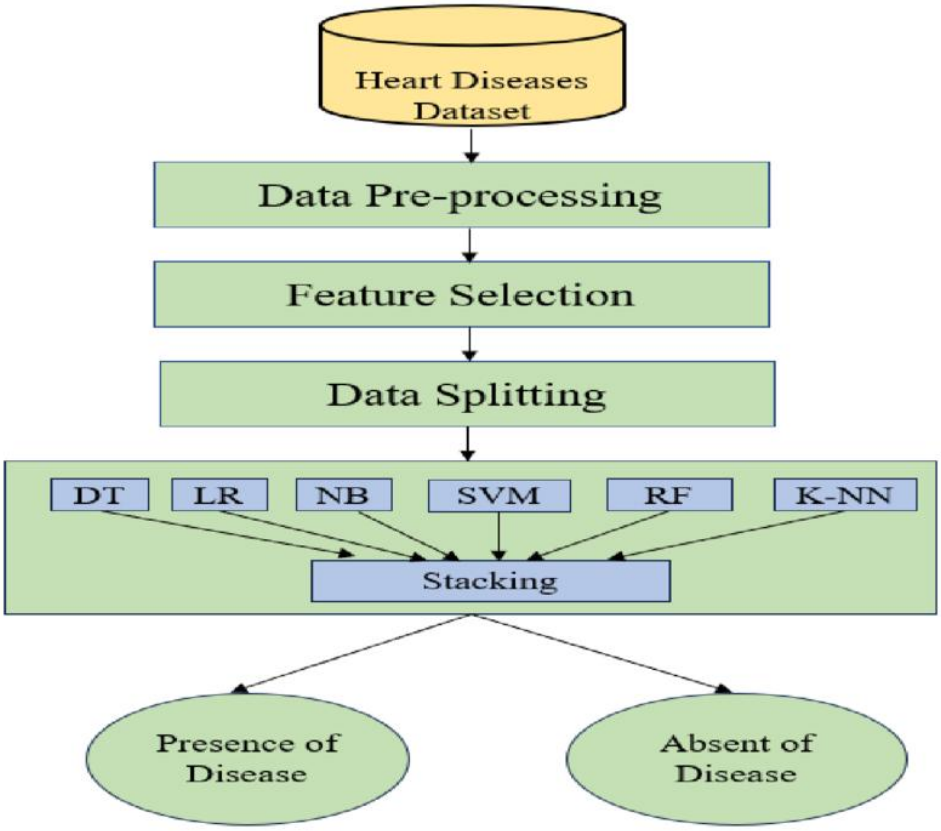


Fig 2.2.1.1 System Architecture

2.2.2. PROPOSED ALGORITHMS /TECHNIQUES

Logistic regression

Logistic regression analysis explores the relationship between a categorical dependent variable and a set of independent variables. The term "logistic regression" is applied when the dependent variable has only two values, such as 0 and 1, or Yes and No. On the other hand, "multinomial logistic regression" is used when the dependent variable has three or more unique values, like Married, Single, Divorced, or Widowed. While the nature of data for the dependent variable differs from that of multiple regressions, the practical application of the procedure remains similar.

Logistic regression serves as a competitor to discriminant analysis in analysing categorical-response variables. Many statisticians logistic regression due to its versatility and suitability for model various situations compared to discriminant analysis. This preference arises from logistic regression's ability to not assume that the independent variables follow a normal distribution, unlike discriminant analysis.

Naive Bayes

The naive Bayes approach is a supervised learning method founded on a simple assumption: it presumes that the presence or absence of one feature of a class is independent of the presence or absence of any other feature. Despite its simplicity, it demonstrates robustness and efficiency comparable to other supervised learning techniques. One explanation often highlighted in the literature is based on representation bias.

The naive Bayes classifier operates as a linear classifier, akin to linear discriminant analysis, logistic regression, or linear support vector machines (SVMs). However, the distinction lies in the method used to estimate the classifier's parameters, known as the learning bias. Although the naive Bayes classifier finds extensive use in the research community due to its ease of programming, parameter estimation simplicity, rapid learning even with large datasets, and reasonably good accuracy compared to other methods, it remains less popular among practitioners seeking practical

results. Researchers appreciate its simplicity and efficacy. However, practitioners often struggle with its interpretability and deployment, as they may not grasp its relevance or utility.

Random forests

Random forests, also known as random decision forests, represent an ensemble learning technique used for classification, regression, and other tasks. They function by constructing numerous decision trees during training. For classification tasks, the output of the random forest is determined by the class selected by the majority of trees. Conversely, for regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests aim to mitigate the issue of decision trees overfitting to their training set.

In general, random forests tend to outperform individual decision trees, although they may have lower accuracy compared to gradient boosted trees. Nonetheless, the performance of random forests can be influenced by the characteristics of the data.

The concept of random decision forests was first introduced in 1995 by Tin Kam Ho, who utilized the random subspace method. This method, as formulated by Ho, serves as an implementation of the "stochastic discrimination" approach to classification initially proposed by Eugene Kleinberg.

Support Vector Machine (SVM)

Support Vector Machine (SVM) represents a discriminant machine learning technique commonly used in classification tasks. It aims to find a discriminant function, based on an independently and identically distributed training dataset, that accurately predicts labels for newly acquired instances. Unlike generative machine learning approaches, which necessitate computations of conditional probability distributions, a discriminant classification function assigns a data point x to one of the classes involved in the classification task. Compared to generative approaches, discriminant methods may be less powerful, particularly in outlier detection scenarios.

However, they require fewer computational resources and less training data, especially in multidimensional feature spaces and when only posterior probabilities are necessary. Geometrically, learning a classifier equates to identifying the equation for a multidimensional surface that optimally separates the different classes in the feature space.

SVM is a discriminant technique that solves convex optimization problems analytically, consistently yielding the same optimal hyperplane parameters. In contrast, genetic algorithms (GAs) and perceptrons, both widely used for classification in machine learning, may produce solutions highly dependent on initialization and termination criteria. With a specific kernel transforming data from the input space to the feature space, SVM training returns uniquely defined model parameters for a given training set, whereas perceptron and GA classifier models vary with each training iteration.

2.3 Proposed Modules

2.3.1 Service Provider

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as View All Heart Disease Data Set Details, Search Heart Disease Data Set Details, Diagnose and Identify Heart Disease, View All Remote Users, Diagnose and Identify Normal User, Diagnose and Identify AbNormal User, View Cholestrol Results, View Heart Beat Results.

2.3.2 View and Authorize Users

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

2.3.3 Remote User

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do

some operations like " Add Heart Disease Data Sets, Search On Heart Disease Details, View Your Profile".

2.3.4 USER CONSTRAINTS

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

- Economical Constraints
- Technical Constraints
- Social Constraints

ECONOMICAL CONSTRAINTS

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

TECHNICAL CONSTRAINTS

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

SOCIAL CONSTRAINTS

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

2.5 HARDWARE REQUIREMENTS

Processor	:	I3 or higher
Speed	:	2.9 GHz
RAM	:	4 GB (min)
Hard Disk	:	160 GB

2.6 SOFTWARE REQUIREMENTS

Operating system	:	Windows 7 Ultimate
Coding Language	:	Python
Back-End	:	Django-ORM
Designing	:	HTML, CSS, JavaScript
Data Base	:	MySQL (WAMP Server)

2.7 Non-Functional Requirements

Non-functional requirements are the constraints that must be adhered during development. They limit what resources can be used and set bounds on aspects of the software's quality.

User Interfaces

The User Interface is a GUI developed using Python.

Software Interfaces

The main processing is done in Java and console application.

Manpower Requirements

5 members can complete the project in 2 – 4 months if they work fulltime on it.

2.8 SDLC Methodologies

Software Development Life Cycle Models and Methodologies

Fig 2.8.1 SDLC

The Software Development Life Cycle (SDLC) is a series of stages that provide a structured approach to the software development process. It encompasses understanding the business requirements, eliciting needs, converting concepts into functionalities and features, and ultimately delivering a product that meets business needs. A proficient software developer should possess adequate knowledge to select the appropriate SDLC model based on project context and business requirements.

Therefore, it is essential to select the right SDLC model tailored to the specific concerns and requirements of the project to ensure its success. To explore more about choosing the right SDLC model, you can follow this link for additional information. Furthermore, to delve deeper into software lifecycle testing and SDLC stages, follow the highlighted links here.

The exploration will cover various types of SDLC models, their benefits, disadvantages, and when to use them. SDLC models can be viewed as tools to enhance product delivery. Therefore, understanding each model, its advantages, disadvantages, and the appropriate usage is crucial to determine which one suits the project context.

Types of Software developing life cycles (SDLC)

- Waterfall Model
- V-Shaped Model
- Evolutionary Prototyping Model
- Spiral Method (SDM)
- Iterative and Incremental Method
- Agile development

Agile Development

The Agile model is an iterative and incremental approach to software development that emphasizes flexibility, customer collaboration, and continuous improvement. It breaks the project into small, manageable increments called iterations or sprints, allowing teams to adapt to changing requirements quickly. Agile is a flexible and adaptive methodology that can be applied to various projects and teams. Its emphasis on collaboration, customer satisfaction, and continuous improvement makes it a popular

choice for many organizations. Agile is widely used in software development, especially for projects with evolving requirements, as it promotes speed, flexibility, and customer satisfaction. Agile is an iterative and incremental software development methodology that emphasizes flexibility, collaboration, and customer satisfaction. Here are some of the stages of Agile Development

1.Project Initiation

2.Sprint Planning

3.Sprint Execution

4.Sprint Review

5.Sprint Retrospective

6.Release Planning

7.Release

The Agile Model can have the multiple phases:

Fig 2.8.2 **Agile Model**

Advantages

- **Faster Time-to-Market:** Deliver working software in short iterations.
- **Improved Collaboration:** Encourages teamwork, communication, and customer involvement.
- **Increased Flexibility:** Adapt to change and new requirements.
- **Enhanced Quality:** Continuous testing and feedback ensure high-quality software.

SYSTEM DESIGN

CHAPTER-3

SYSTEM DESIGN

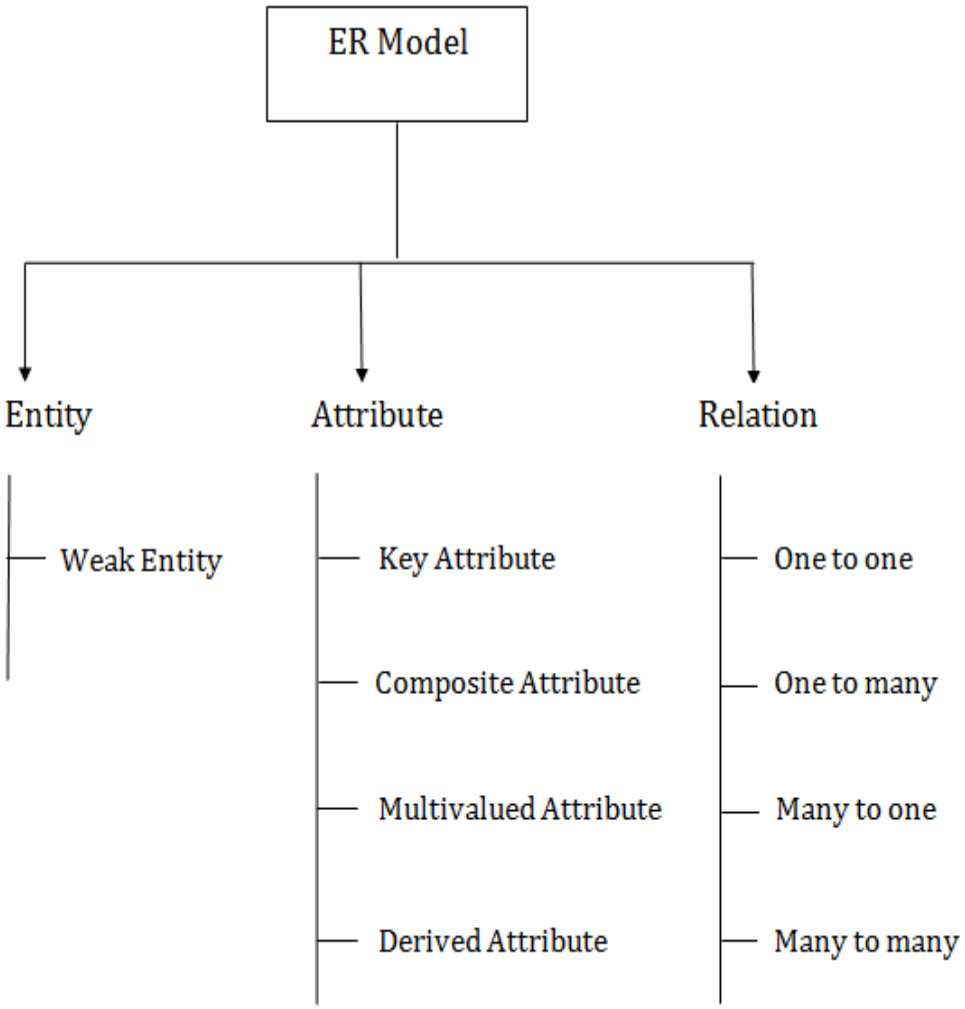
3.1 Data Design

An Entity-Relationship (ER) model illustrates the structure of a database using a visual representation known as an Entity-Relationship Diagram (ER Diagram). This model serves as a blueprint for designing the database schema and capturing the relationships between different entities and attributes.

The ER model provides a systematic approach to organizing and conceptualizing the data within a database system. It represents entities as well as the relationships between them, helping to clarify how data elements are connected and organized**.**

3.1.1 ER model

- The Emergency Room model corresponds to an Entity-Relationship model, serving as a high-level representation of data structures. It is utilized to illustrate the data components and relationships within a defined system.
- It establishes a structured framework for the database. Moreover, it provides a straightforward and easily understandable perspective on the data.
- In Entity-Relationship model, the organizational database structure is depicted through a design known as an Entity-Relationship diagram.
- For instance, consider designing a school database. An educational record could be represented as an entity with attributes such as name, ID, age, etc. Similarly, the address could be another entity with attributes like city, street name, zip code, etc., and there would be a relationship between them.



Component of ER Diagram

1. Entity

A substance may be anything, class, individual or spot. In the ER frame, a substance can be tended to as square shapes.

Think about a relationship as a delineation chief, thing, specialist, office, etc.

Powerless Entity

A substance that depends upon another component called afrail substance. The frail element contains no critical trait of its own. The feeble substance is addressed by a twofold square shape.

Characteristic

The quality is utilized to depict the property of a section. Obscure is utilized to address a quality.

For example, id, age, contact number, name, etc. can be attributes of a student.

a. Key Attribute

The key quality is used to address the essential ascribes of a substance. It tends to a fundamental key. The key property is tended to by a circle with the text underlined.

b. Composite Attribute

A property that made from various attributes is known as a composite quality. The composite trademark is tended to by an oval, and those circles are related with a circle**.**

c. Multivalued Attribute

A quality can have more than one worth. These qualities are known as a multivalued property. The twofold oval is used to address multivalued property. For example, a student can have more than one phone number.

d. Determined Attribute

A property that can be gotten from another quality is known as a decided attribute. It will in general be tended to by a ran circle.

For example, a singular's age changes long term and can be gotten from one more quality like Date of birth**.**

3. Relationship

A relationship is used to depict the connection between substances. Important stone or rhombus is utilized to address the relationship. Sorts of relationship are as per the following

a. One-to-One relationship

At the point when just a single instance of a component is connected with the relationship, then it is known as facilitated relationship. For instance, A female can wed to one male, and a male can wed to one female**.**

b. One-to-many relationship

Exactly when simply a solitary illustration of the substance on the left, and more than one event of a component on the right associates with the relationship then this is known as a one-to-various connections.

For example, Scientist can envision various manifestations, but the improvement is done by the really express analyst**.**

c. Many-to-one relationship

Exactly when more than one event of the component on the left, and simply a solitary event of a substance on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student login for only a solitary course, but a course can have various students.

d. Many-to-many relationship

At the point when more than one event of the substance on the left, and more than one event of a component on the right associates with the relationship then it is known as a many-to-various connections.

For example, Employee can allot by numerous exercises and project can have various specialists.

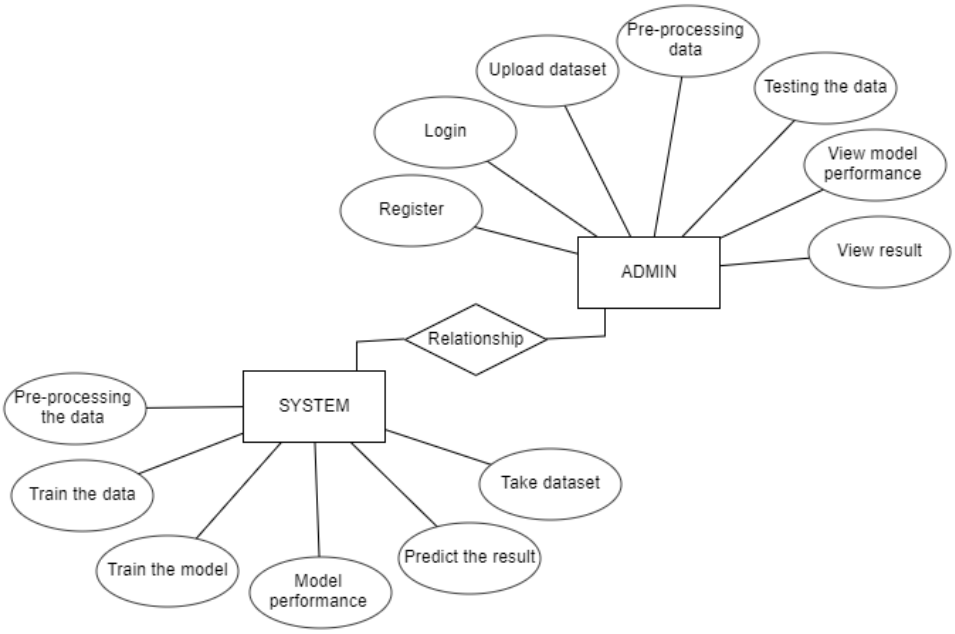


Fig 3.1.1.1 ER Diagram

3.2. Data Dictionary

A Data Dictionary compiles names, definitions, and attributes concerning data elements utilized or stored within a database, information system, or part of a research project. It delineates the meanings and functions of data elements within the context of a project and offers guidance on understanding, recognizing meanings, and description. Additionally, a Data Dictionary offers metadata about data elements, aiding in defining the scope and attributes of data elements, as well as the guidelines for their usage and application.

Name	Type	Collation	Attributes	Null	Default
User name	Varchar (30)	Latin_swedish_ci		No	None
email	Varchar (30)	Latin_swedish_ci		No	
pswd	Varchar (30)	Latin_swedish_ci		No	
phone no	Varchar (30)	Latin_swedish_ci		No	
country	Varchar (30)	Latin_swedish_ci		No	
State	Varchar(30)	Latin_swedish_ci		No	
City	Varchar(30)	Latin_swedish_ci		No	

Table no.3.2 Data Dictionary

User name	email	pswd	phno	state	city
Maaju	Maajushaik18@gmail.com	********	6281407028	AP	Kadapa
Ayesha	Ayeshashaik23@gmail.com	********	8790557392	AP	Kadapa
Sravs	Sravk23@gmail.com	********	7780490892	AP	Pld
Bunny	Saik05@gmail.com	********	9849466191	AP	Kpl

Table no 3.2.1 Data table

3.3 Normalization

Normalization is the primary method for optimizing data in a database to fulfill two essential criteria:

Data dependencies are logical, ensuring that all related data items are stored together. Normalization is crucial for various reasons, primarily

because it enables databases to occupy minimal disk space, resulting in enhanced performance.

Normalization is also referred to as data standardization.

The three primary types of normalization are outlined below. Note: "NF" stands for "normal form."

First typical structure (1NF)

Tables in 1NF should comply with certain standards:

1. Every cell should contain just a solitary (nuclear) esteem.
2. Each part in the table ought to be astoundingly named.
3. All characteristics in a part ought to connect with a comparative region.

Username	**Password**
John	********
Princess	********
Tom	********
Claire	********
Robert	********

Table No 3.3.1 1NF

Second Typical structure (2NF)

Tables in 2NF ought to be in 1NF and not have any most of the way dependence (e.g., each non-prime quality ought to be dependent upon the table's fundamental key).

Received Data through IOT	Pswd	Login
11	********	Sign_up
12	********	Sign_up
13	********	Sign_up
14	********	Sign_up
15	********	Sign_up

Table no3.3.2 2NF

Third ordinary structure (3NF)

Tables in 3NF ought to be in 2NF and have no transitive reasonable circumstances on the fundamental key. The going with two NFs furthermore exists anyway are only here and there used:

USERDETAILS

NAME	EMAIL	STATE	CITY	COUNTRY
Vijay	vijay@gmail.com	AP	RZP	INDIA
Vinod	vinod@gmail.com	AP	RZP	INDIA
Ramu	Ramu@gmail.com	AP	RZP	INDIA
Vishnu	vishnu@gmail.com	AP	RZP	INDIA

Table no 3.3.3 User Details

USER DETAILS

USER NAME	PASSSWORD	LOGIN
server	********	Sign_up
Vijay	********	Sign_up

Table no3.3.4 User details

Boyce-Codd Normal Form (BCNF)

Normalization is a critical process in database management aimed at organizing tables to minimize anomalies and ensure data integrity. It follows a series of stages known as normal forms. These normal forms help structure tables efficiently and reduce redundancy and inconsistency in data.

Unnormalized Form (UNF)

The initial state of a table where data is not organized according to any specific rules.

First Normal Form (1NF)

In 1NF, each column contains atomic values, and there are no repeating groups or arrays within a row.

Second Normal Form (2NF)

2NF requires that every non-key attribute be fully functionally dependent on the primary key.

Third Normal Form (3NF)

In 3NF, no transitive dependencies should exist, meaning that non-key attributes should not depend on other non-key attributes.

Elementary Key Normal Form (EKNF)

EKNF is a further refinement of 3NF, emphasizing the use of elementary keys.

Boyce-Codd Normal Form (BCNF)

BCNF addresses anomalies that may arise when multiple candidate keys exist. It requires that for every non-trivial functional dependency (X → Y), X must be a super key.

Fourth Normal Form (4NF)

To achieve 4NF, a table must be in BCNF and should not have multi-valued dependencies**.**

Essential Tuple Normal Form (ETNF)

ETNF is a condition where each attribute in a tuple is essential to the understanding of the tuple itself**.**

Normal Form	Description
1NF	An alliance is in 1NF enduring it contains an atomic worth.
2NF	An association will be in 2NF expecting it is in 1NF and all non-key credits are totally down to earth ward on the fundamental key.
3NF	An alliance will be in 3NF enduring it is in 2NF and no change dependence exists.
BCNF	A more grounded importance of 3NF is known as Boyce Codd's common design.
4NF	An association will be in 4NF expecting it is in Boyce Codd's commonplace construction and has no multi-regarded dependence.
5NF	An association is in 5NF. In case it is in 4NF and contains no join dependence, joining should be lossless.

Benefits of Normalization

Reduction of data redundancy: Normalization helps eliminate redundant data by organizing it efficiently across tables. Improved overall database organization: By structuring data according to normalization rules, databases become more organized and easier to manage.

Data consistency within the database: Normalization ensures that data remains consistent across tables, reducing the risk of inconsistencies.

More flexible database design: Normalization allows for more flexibility in database design, making it easier to accommodate changes and updates. Upholds the principle of data integrity: Normalization promotes data integrity by minimizing anomalies and ensuring accurate representation of data relationships.

Disadvantages of Normalization

Careless decomposition: If normalization is done without a clear understanding of user requirements, it can lead to excessive decomposition and unnecessary complexity in the database design.

Decreased performance: As tables are normalized to higher normal forms such as 4NF or 5NF, it may lead to decreased performance due to increased join operations and complexity in querying the database**.**

UML INTRODUCTION

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntax, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective.

UML is specifically constructed through two different domains they are:

UML Analysis modeling, this focuses on the user model and structural model views of the system.

UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Once the analysis stage is completed, the next stage is to determine in broad outline form how the problem might be solved. During system design, we are beginning to move from the logical to physical level.

System design involves architectural and detailed design of the system. Architectural design involves identifying software components, decomposing them into processing modules and conceptual data structures, and specifying the interconnections among components.

Detailed design is concerned with how to package processing modules and how to implement the processing algorithms, data structures and interconnections of standard algorithms, invention of new algorithms, and design of data representations and packaging of software products.

Two kinds of approaches are available:

1.Top down approach

2.Bottom up approach

Design of Code

Since information systems projects are designed with space, time and cost saving in mind, coding methods in which conditions, words, ideas or control errors and speed the entire process. The purpose of the code is to facilitate the identification and retrieval of the information. A code is an ordered collection of symbols designed to provide unique identification of an entity or an attribute.

Design of Input

Design of input involves the following decisions

1.Input data

2.Input medium

The way data should be arranged or coded

Validation needed to detect every step to follow when error occurs

The input controls provide ways to ensure that only authorized users access the system guarantee the valid transactions, validate the data for accuracy and determine whether any necessary data has been omitted. The primary input medium chosen is display. Screens have been developed for input of data using HTML. The validations for all important inputs are taken care of through various events using JSP control.

Design of Output

Design of output involves the following decisions

1.Information to present

2.Output medium

Output layout

Output of this system is given in easily understandable, user-friendly manner, Layout of the output is decided through the discussions with the different users**.**

Design of Control

The system should offer the means of detecting and handling errors.

- Input controls provides ways per
- Valid transactions are only acceptable
- Validates the accuracy of data

Ensures that all mandatory data have been captured

All entities to the system will be validated. And updating of tables is allowed for only valid entries. Means have been provided to correct, if any by change incorrect entries have been entered into the system they can be edited.

3.3. UML DIAGRAMS

Why We Use UML in projects?

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs. Simply, Systems design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

In the project four basic UML diagrams have been explained among the following list:

- Class Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- Collaboration Diagram
- Deployment Diagram
- State Chart Diagram
- Component Diagram

Class Diagram

A Class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

This is one of the most important of the diagrams in development. The diagram breaks the class into three layers. One has the name, the second describes its attributes and the third its methods. A padlock to left of the name represents the private attributes. The relationships are drawn between the classes. Developers use the Class Diagram to develop the classes. Analyses use it to show the details of the system.

Architects look at class diagrams to see if any class has too many functions and see if they are required to be split.

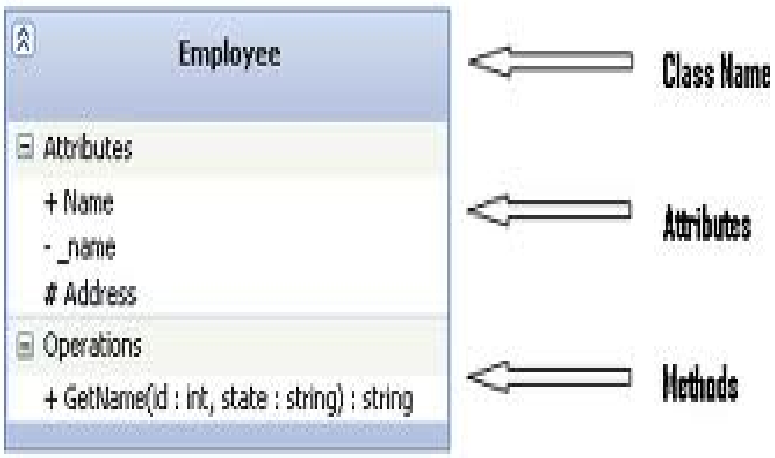


Fig 3.1: Class Diagram

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. Use cases are used during

requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from the external point of view. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system

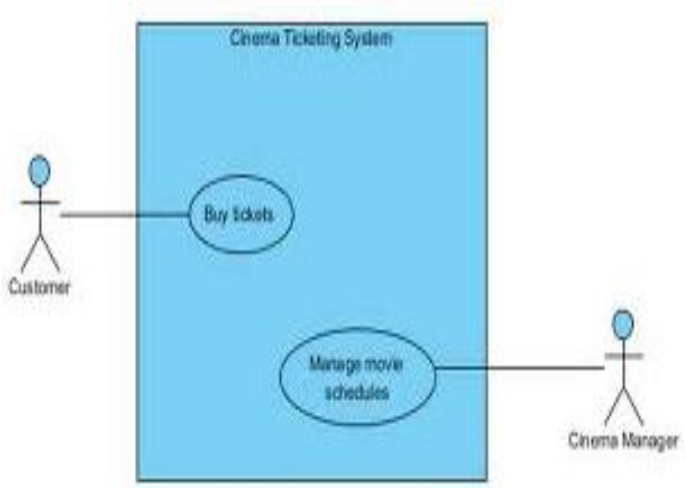


Fig.3.2: Use Case Diagram

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. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.

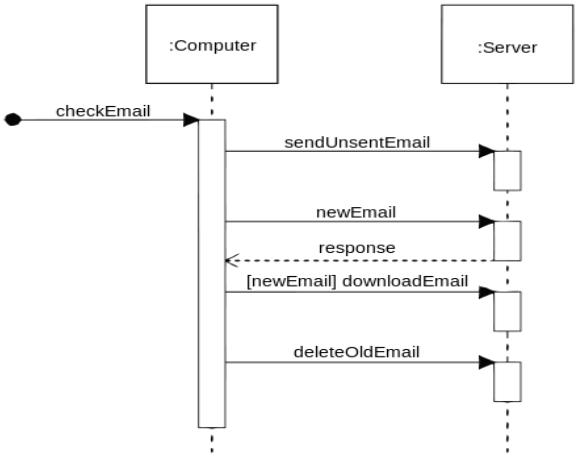


Fig.3.3: Sequence Diagram

Activity Diagram

Activity diagrams are a loosely defined diagram technique for showing 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.

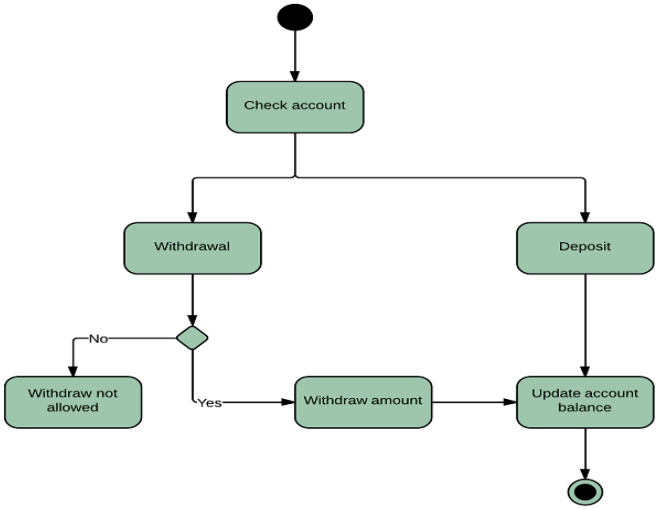


Fig.3.4: Activity Diagram

Collaboration Diagram

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior.

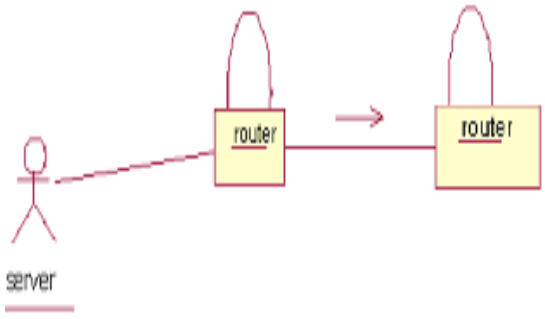


Fig.3.5: Collaboration Diagram

Deployment Diagram

A Deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected e.g. JDBC, REST

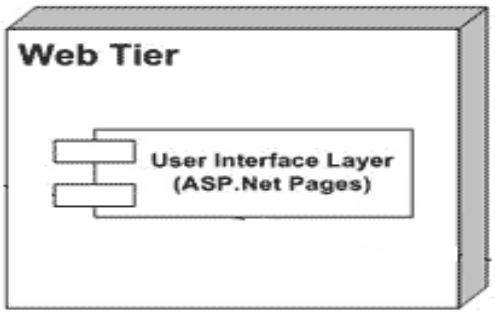


Fig.3.6: Deployment Diagram

State Chart Diagram

A State diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

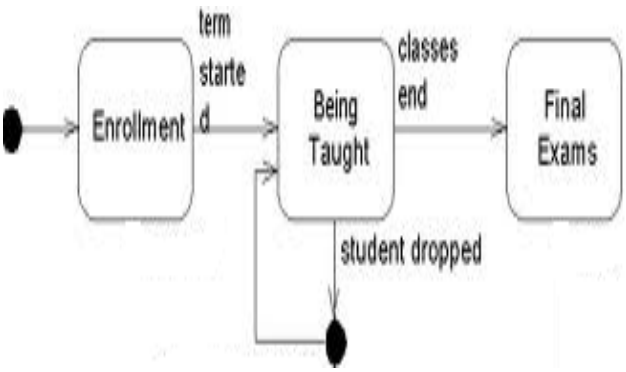


Fig.3.7: State Chart Diagram

Component Diagram

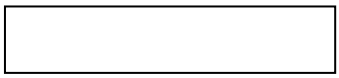
In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Fig.3.8: Component Diagram

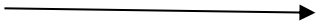
DATA FLOW DIAGRAM

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

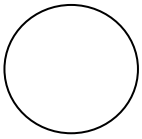
DFD NOTATIONS



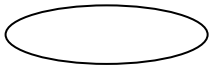
Define source and destination data.



Shows path of the data flow.



To represent a process transforms or modifies the Data



To represent an attribute

Data Store

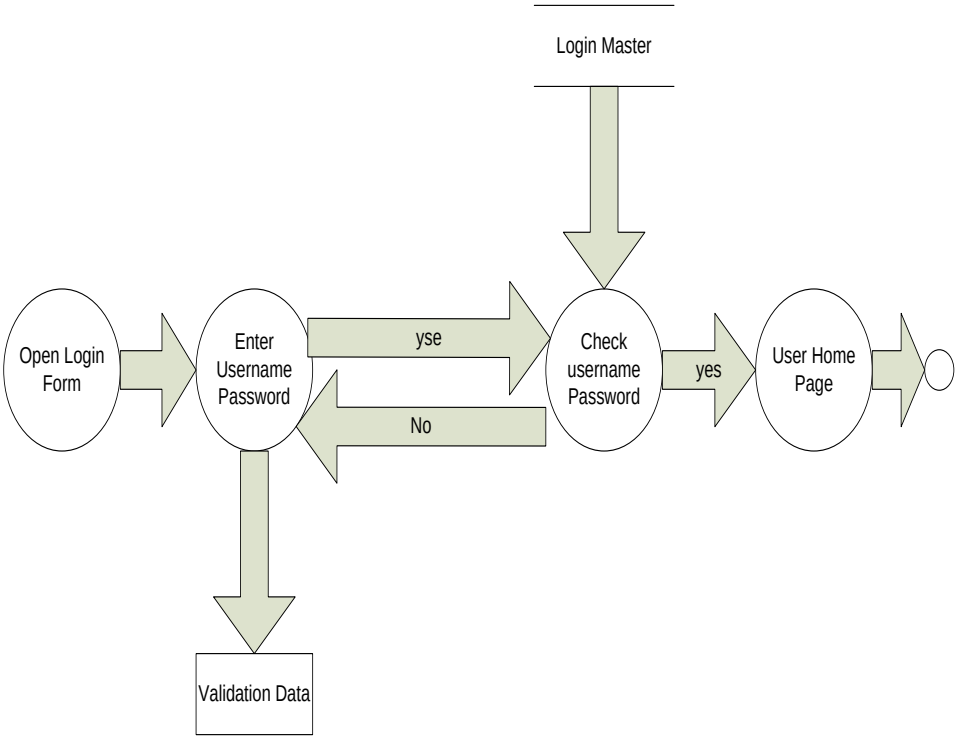


Fig.3.9: Data Flow Diag

UML DIAGRAMS

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

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

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

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

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

GOALS:

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

1.Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.

2.Provide extendibility and specialization mechanisms to extend the core concepts.

3.Be independent of particular programming languages and development process.

4.Provide a formal basis for understanding the modeling language.

5.Encourage the growth of OO tools market.

6.Support higher level development concepts such as collaborations, frameworks, patterns and components.

3.3.1. USE CASE DIAGRAM

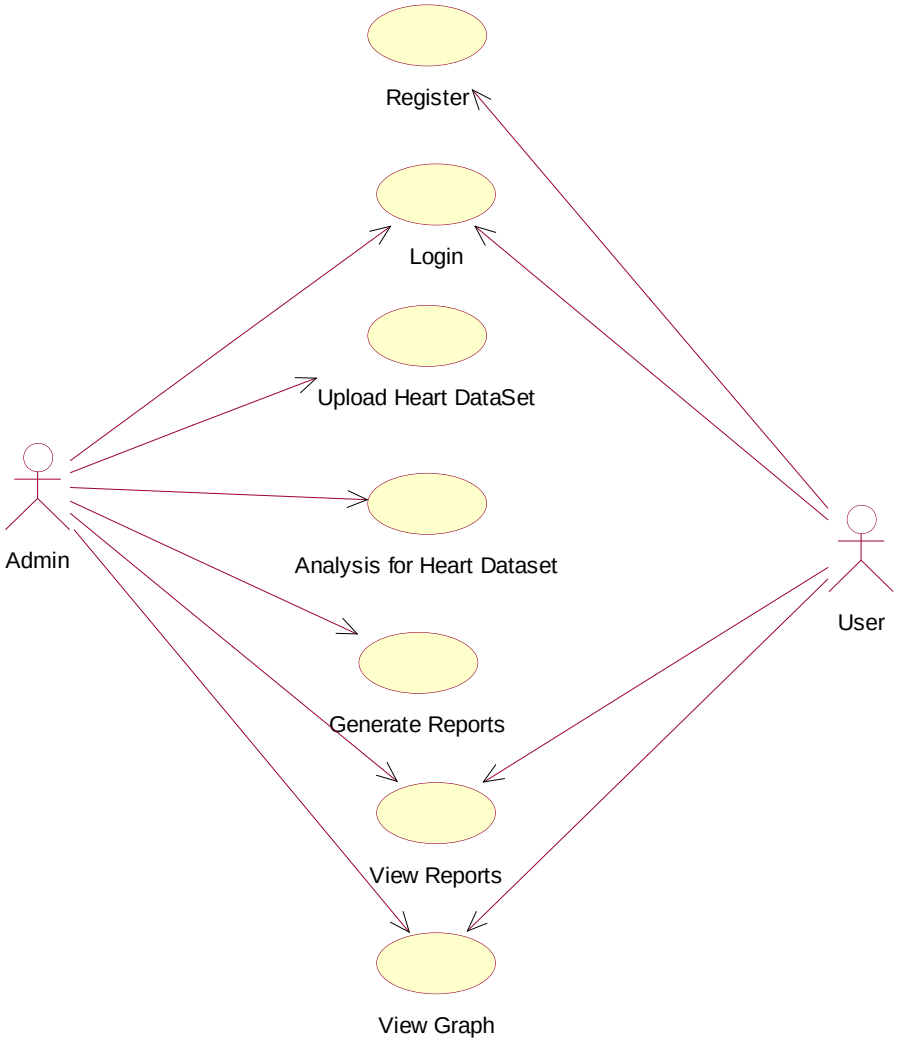


Fig.3.1.: Use case Diagram for overall project

3.3.2. CLASS DIAGRAM

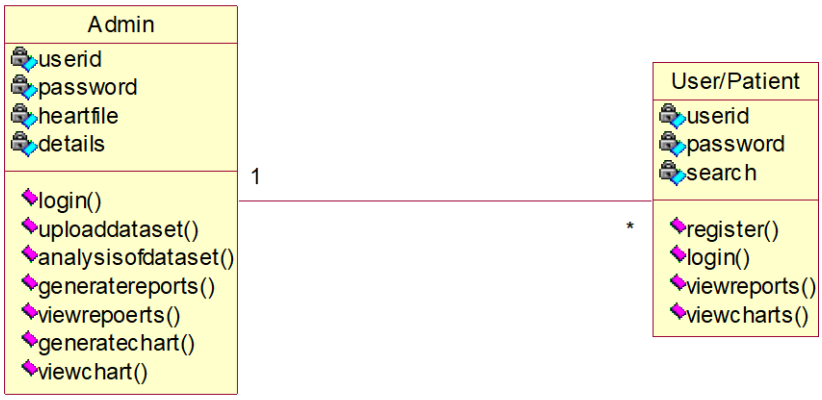


Fig 3.2 Class Diagram

3.3.3. SEQUENCE DIAGRAM

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

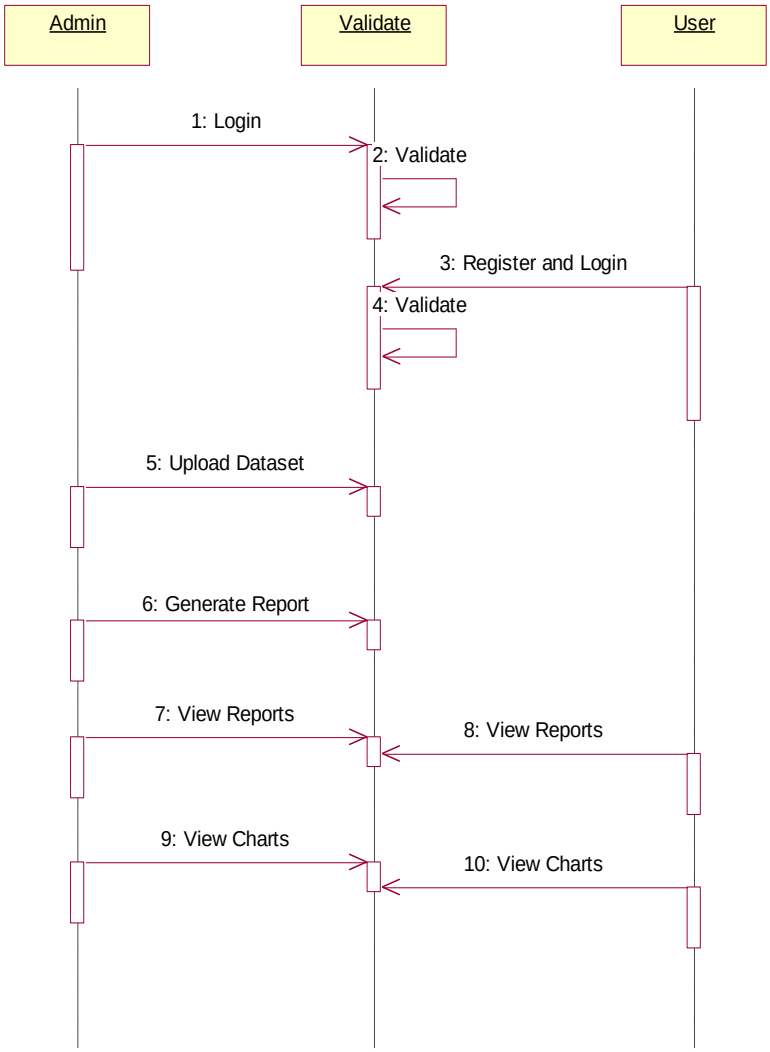


Fig.3.3: Sequence Diagram for Overall Project

3.3.4. ACTIVITY DIAGRAM

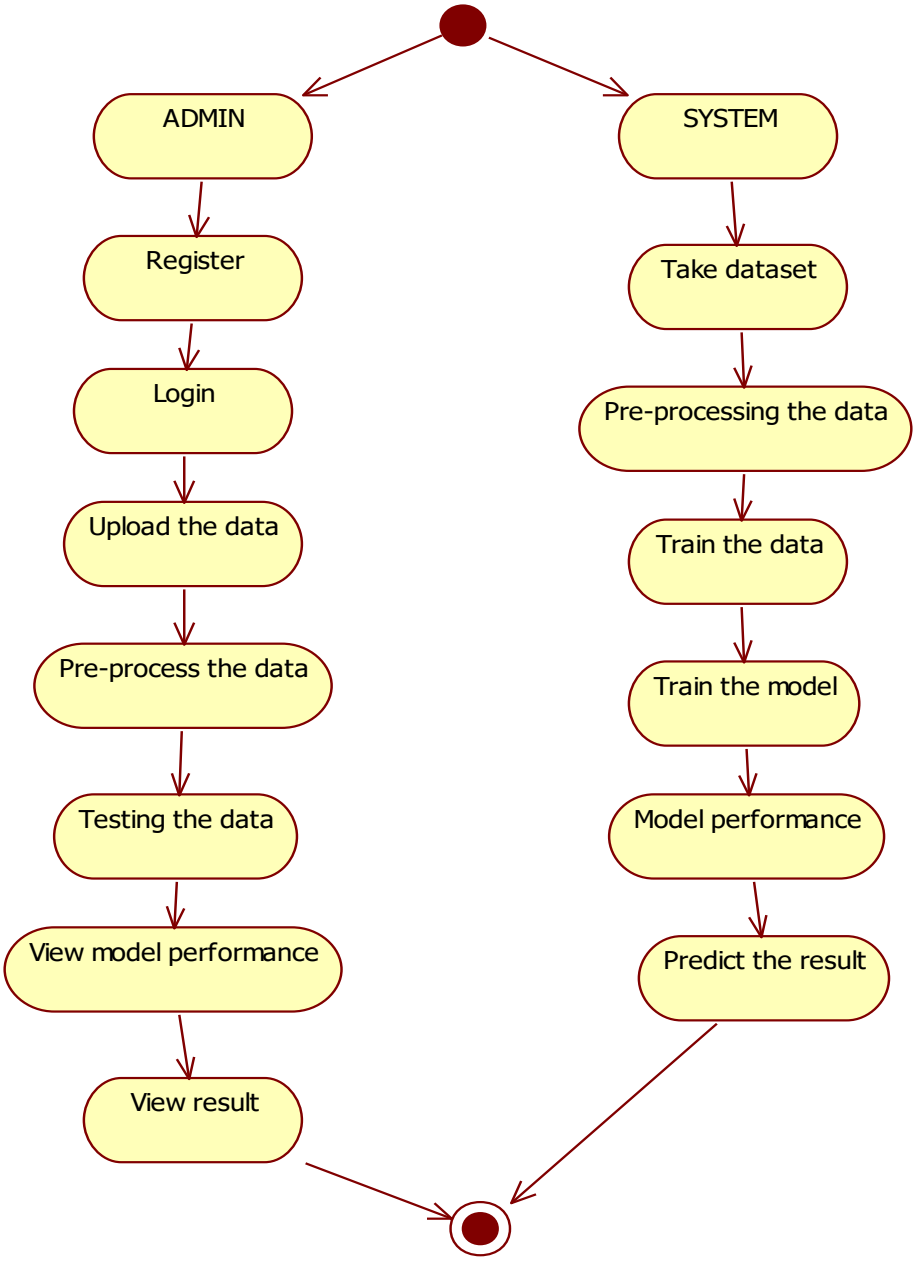


Fig.3.4: Activity Diagram

TESTING

CHAPTER-4

TESTING

4.1 Testing Methodologies

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer**.**

Objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a probability of finding an as yet
- undiscovered error.
- A successful test is one that uncovers an undiscovered error.
- These above objectives imply a dramatic change in view port.
- Testing cannot show the absence of defects, it can only show that software errors are present.

Test Case Design

Any engineering product can be tested in one of two ways**:**

White Box Testing

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis Path Testing

- Flow graph notation
- Cyclomatic Complexity

Deriving test cases Control Structure Testing

- Condition testing
- Data flow testing
- Loop testing

Black Box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing
- Graph matrices

TESTING STRATEGIES

A Strategy for software testing integrates software test cases into a series of well planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers he set of activities that ensure that the software that has been built is traceable to customer's requirements.

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries

of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

Integration Testing

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

Top-Down Integration

Top down integrations is an incremental approach for construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control program. Modules subordinate to the main program are incorporated in the structure either in the breath-first or depth-first manner.

Bottom-up Integration

This method as the name suggests, begins construction and testing with atomic modules i.e., modules at the lowest level. Because the modules are integrated in the bottom up manner the processing required for the modules subordinate to a given level is always available and the need for stubs is eliminated.

Regression Testing

In this contest of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagate unintended side effects.

4..2.3 Validation Testing

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

Reasonable expectation is defined in the software requirement specification – a document that describes all user-visible attributes of the software. The specification contains a section titled “Validation Criteria”. Information contained in that section forms the basis for a validation testing approach.

Validation Test Criteria

Software validation is achieved through a series of black-box tests that demonstrate conformity with requirement. A test plan outlines the classes of tests to be conducted, and a test procedure defines specific test cases that will be used in an attempt to uncover errors in conformity with requirements. Both the plan and procedure are designed to ensure that all functional requirements are satisfied, all performance requirements are achieved, documentation is correct and human-engineered; and other requirements are met.

After each validation test case has been conducted, one of two possible conditions exists: (1) The function or performance characteristics conform to specification and are accepted, or (2) a deviation from specification is uncovered and a deficiency list is created. Deviation or error discovered at this stage in a project can rarely be corrected prior to scheduled completion. It is often necessary to negotiate with the customer to establish a method for resolving deficiencies.

Alpha and Beta Testing

It is virtually impossible for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted. Strange combination of data may be regularly used; and output that seemed clear to the tester may be unintelligible to a user in the field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements. Conducted by the end user rather than the system developer, an acceptance test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a

period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time.

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

System Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

Security Testing

Attempts to verify the protection mechanisms built into the system**.**

4..2.6 Performance Testing

This method is designed to test runtime performance of software within the context of an integrated system.

4..3. TEST CASES

S. No.	TEST CASES	INPUT	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	User Registration	Enter all fields	User gets registered	Registration is successful	pass
2	User Registration	if user miss any field	User not registered	Registration is un successful	fail
3	Admin Login	Give the user name and password	Admin home page should be opened	Admin home Page has been opened	Pass
4	Upload HEART Dataset	Test whether the HEART Dataset is uploaded or not into the system	If HEART Dataset is not uploaded	We cannot do further operations	HEART Dataset uploaded we will do further operations
5	Preprocess Dataset	Verify the Dataset is Per – processed or not	Without loading the dataset	We cannot Per-processing Dataset	We Can Pre-process Dataset successfully
6	Run SVM algorithm	Verify the SVM algorithm will run or not	Without training model	We cannot run SVM algorithm	We can run SVM algorithm

Table 4.1: Test Case Results

IMPLEMENTATION

CHAPTER-5

IMPLEMENTATION

5.1 Working Model Installation Procedure

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 library 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. This further aids the readability of the code.

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 Cheat sheet. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

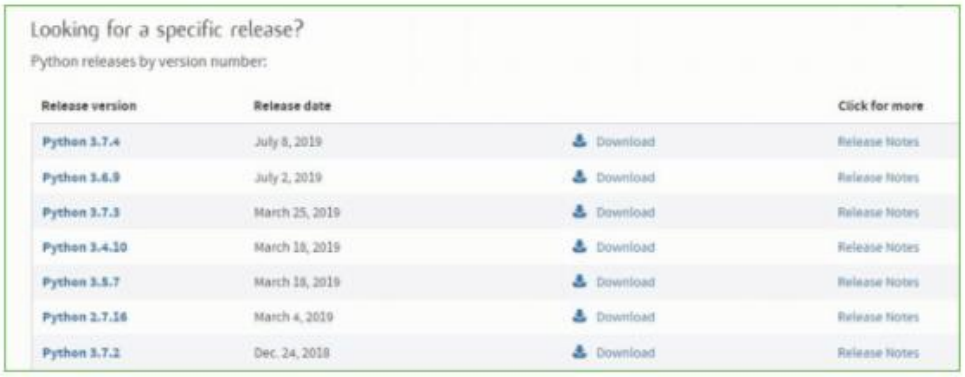
Download the Correct version into the system

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.

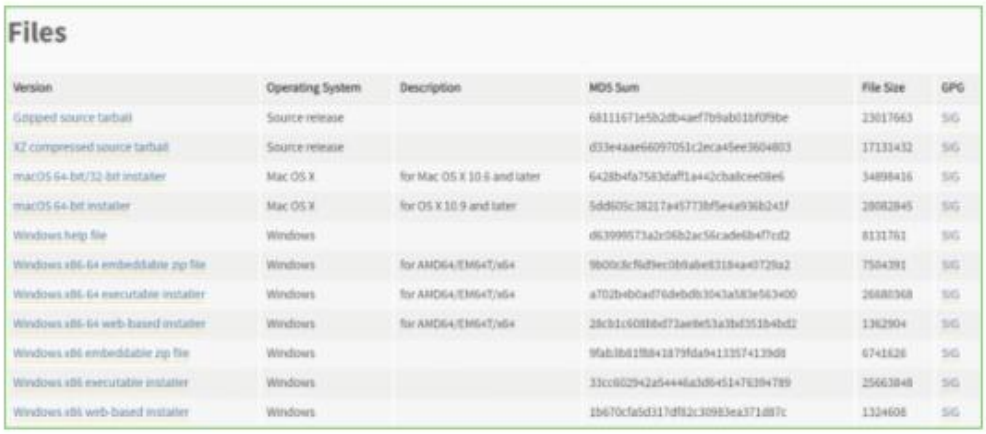
Step 2: Click on the Download Tab.

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



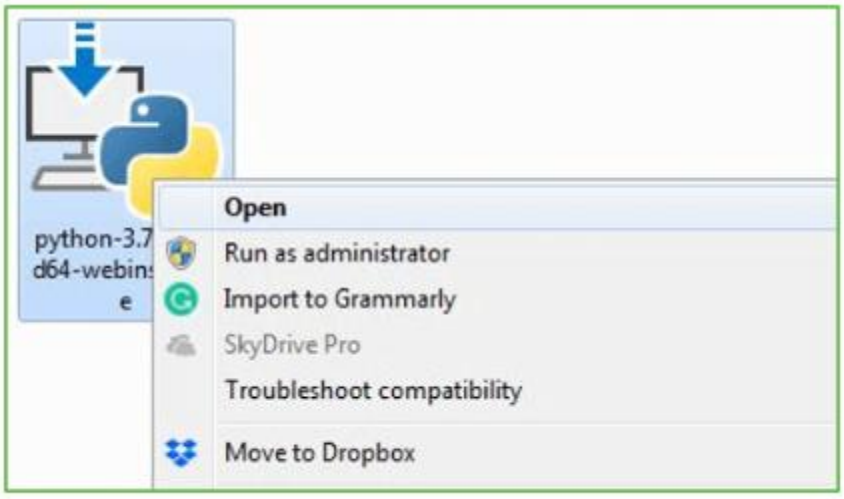
Step 4: Scroll down the page until you find the Files option**.**

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



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.

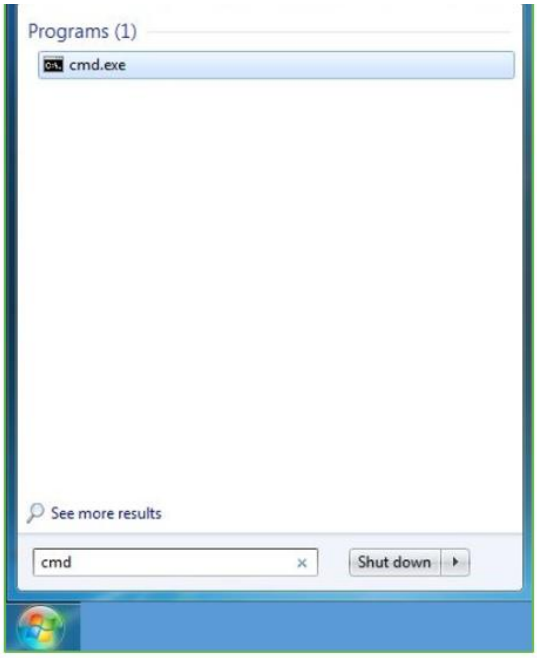
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

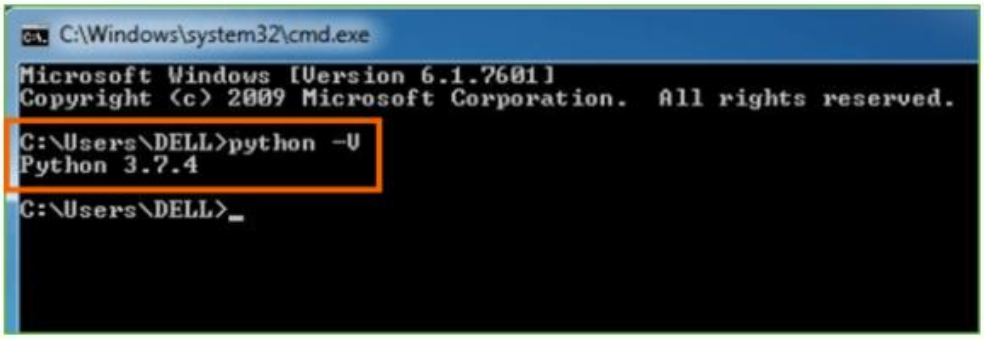
Step 1: Click on Start

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.



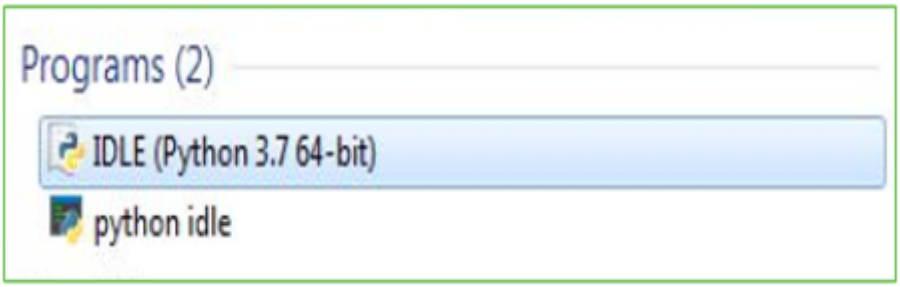
Step 5: You will get the answer as 3.7.4

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

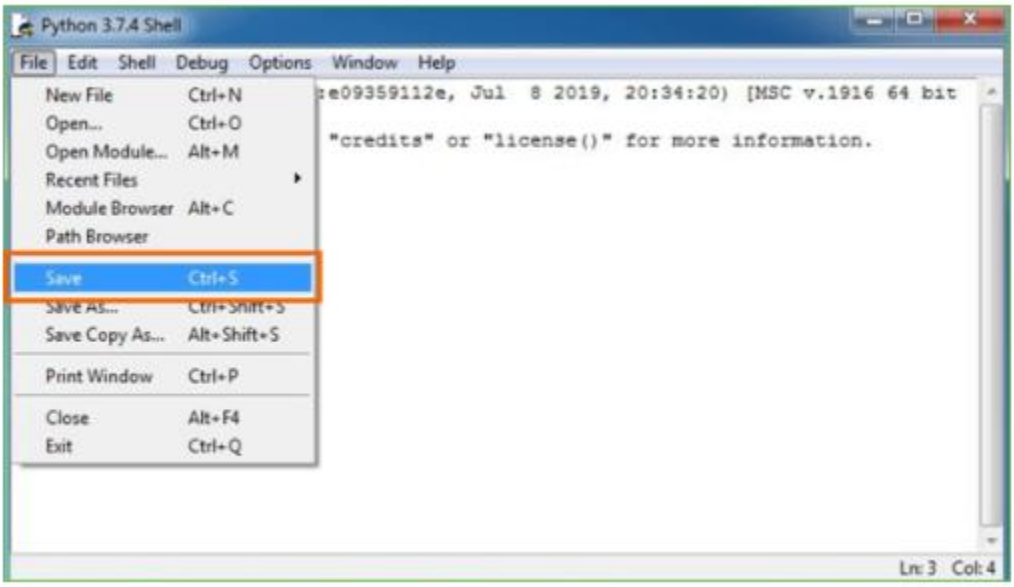
Step 1: Click on Start

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



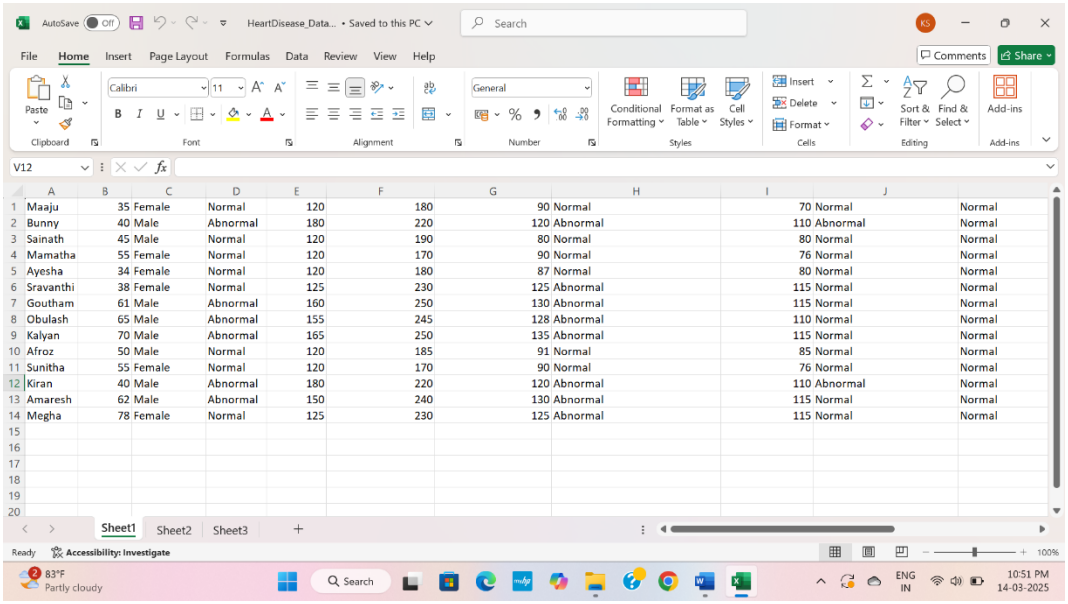
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

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.

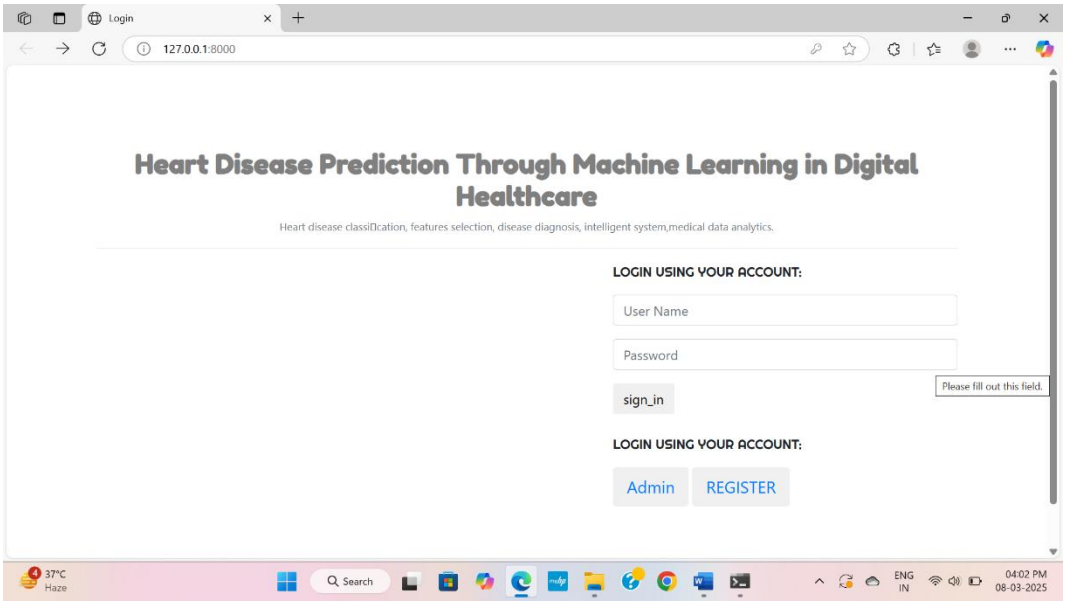
5.2 Sample Screens & Reports



	A	B	C	D	E	F	G	H	I	J	
1	Maaju	35	Female	Normal	120	180	90	Normal	70	Normal	Normal
2	Bunny	40	Male	Abnormal	180	220	120	Abnormal	110	Abnormal	Normal
3	Sainath	45	Male	Normal	120	190	80	Normal	80	Normal	Normal
4	Mamatha	55	Female	Normal	120	170	90	Normal	76	Normal	Normal
5	Ayesha	34	Female	Normal	120	180	87	Normal	80	Normal	Normal
6	Sravanthi	38	Female	Normal	125	230	125	Abnormal	115	Normal	Normal
7	Goutham	61	Male	Abnormal	160	250	130	Abnormal	115	Normal	Normal
8	Obulash	65	Male	Abnormal	155	245	128	Abnormal	110	Normal	Normal
9	Kalyan	70	Male	Abnormal	165	250	135	Abnormal	115	Normal	Normal
10	Afroz	50	Male	Normal	120	185	91	Normal	85	Normal	Normal
11	Sunitha	55	Female	Normal	120	170	90	Normal	76	Normal	Normal
12	Kiran	40	Male	Abnormal	180	220	120	Abnormal	110	Abnormal	Normal
13	Amaresh	62	Male	Abnormal	150	240	130	Abnormal	115	Normal	Normal
14	Megha	78	Female	Normal	125	230	125	Abnormal	115	Normal	Normal

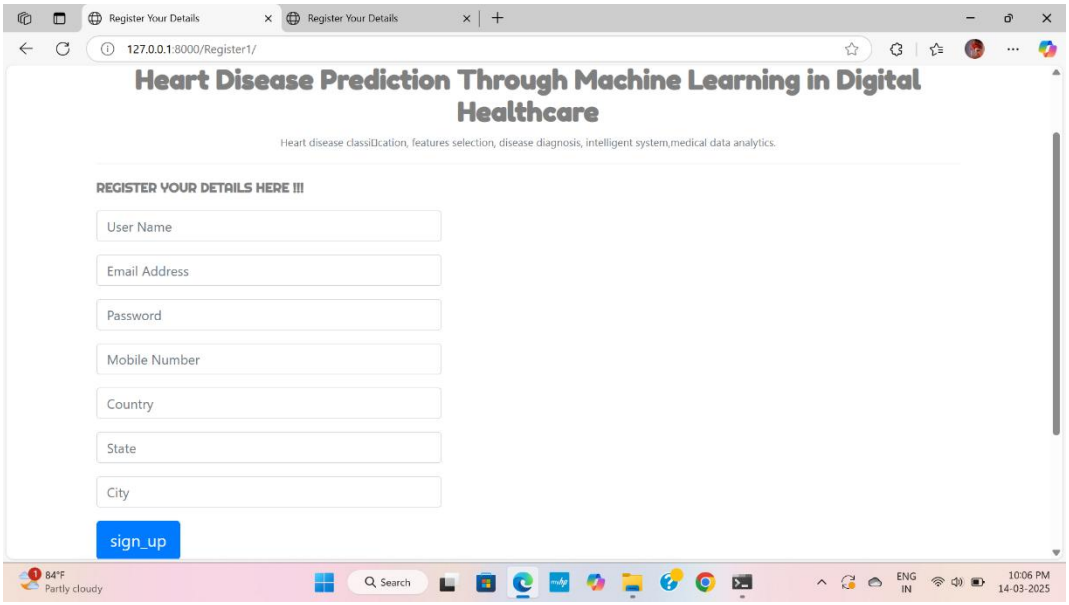
Screen 5.2.1 Showing Dataset

Description: The above page shows the Patient Details Dataset.



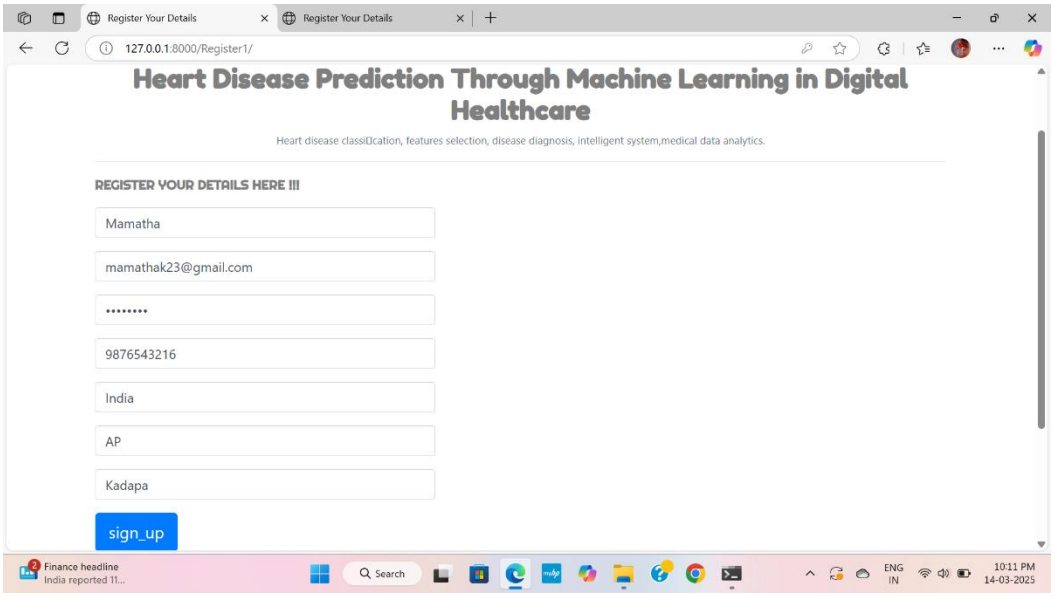
Screen 5.2.2 Home page

Description: In above screen click on 'New User Register Here' link to get below screen.



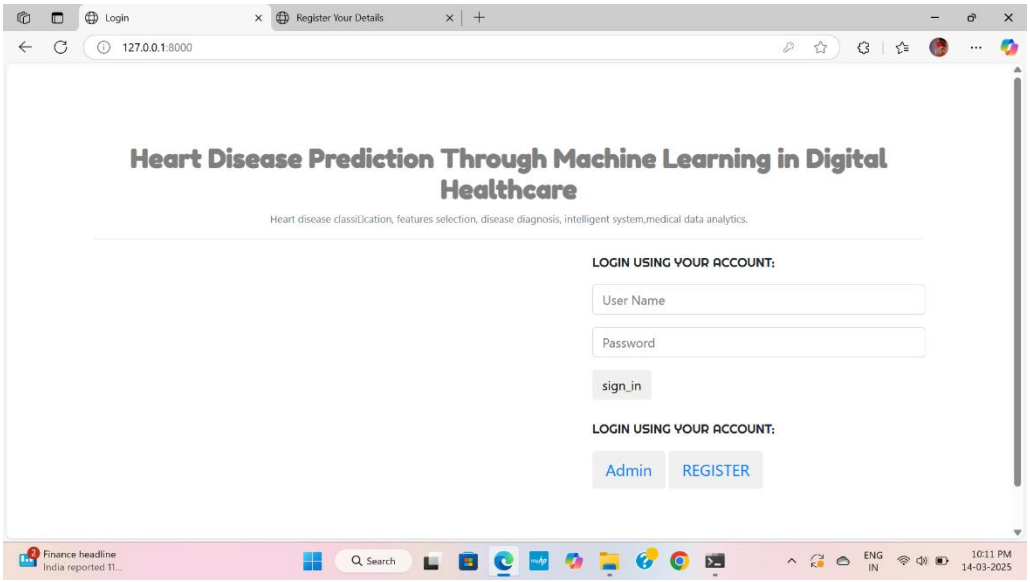
Screen 5.2.3 Registration Page

Description: In above screen user is register and enter the new user Details.



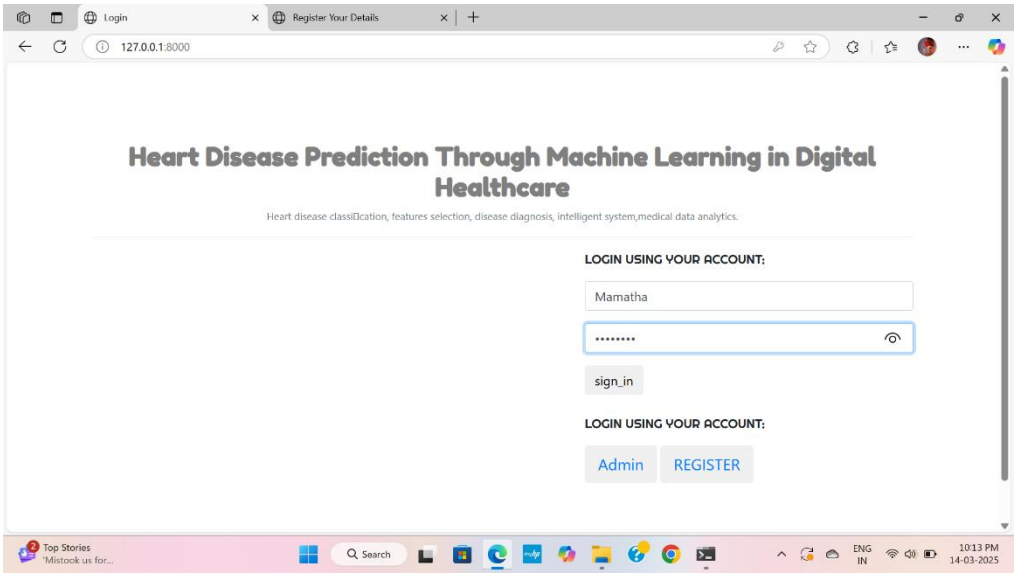
Screen 5.2.4 User Details

Description: In above screen user details was entered now click on ”User Login” link to get below screen.



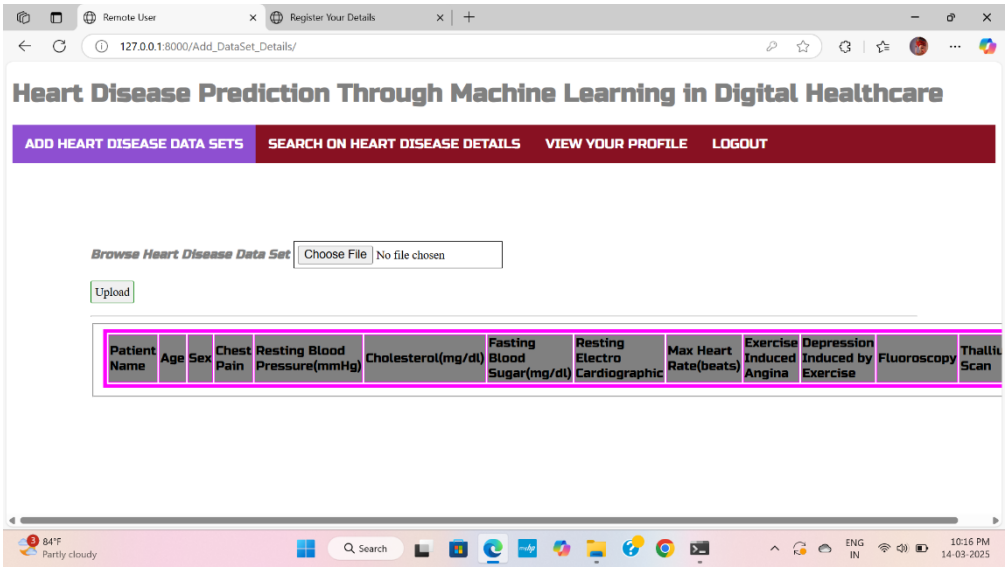
Screen 5.2.5 Login Page

Description: In above screen user is Login screen and enter the user Details.



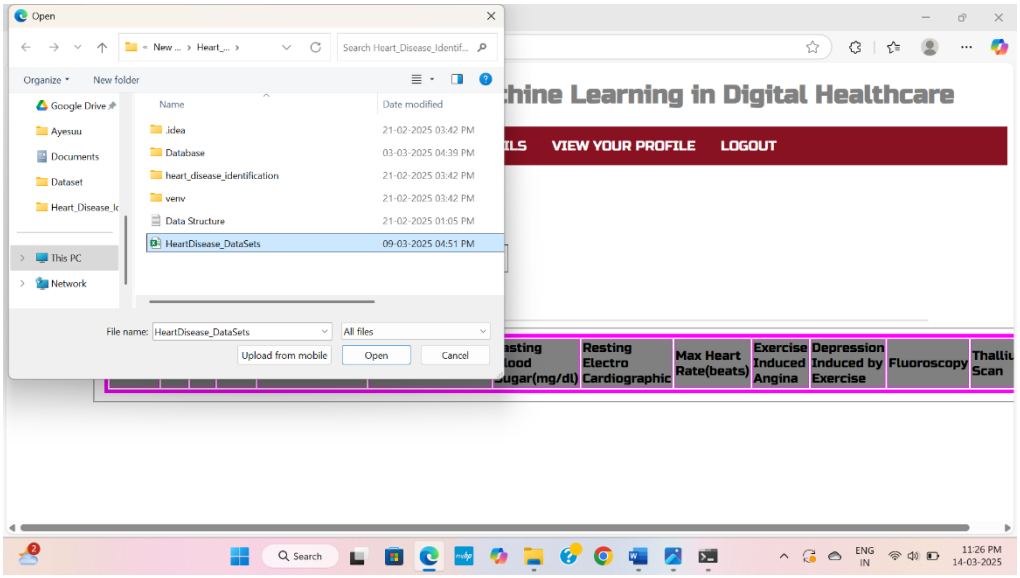
Screen 5.2.6 User Details Page

Description: In above screen user is login and after login will get below screen.



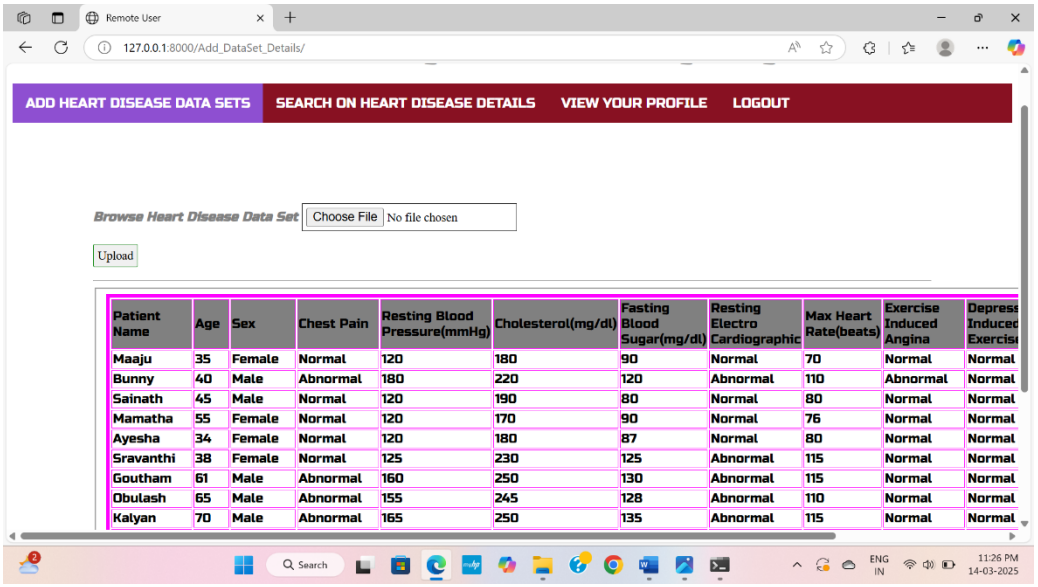
Screen 5.2.7 Home Page

Description: The above screen shows the Details of the Patient where we choose, enter and Upload the File.



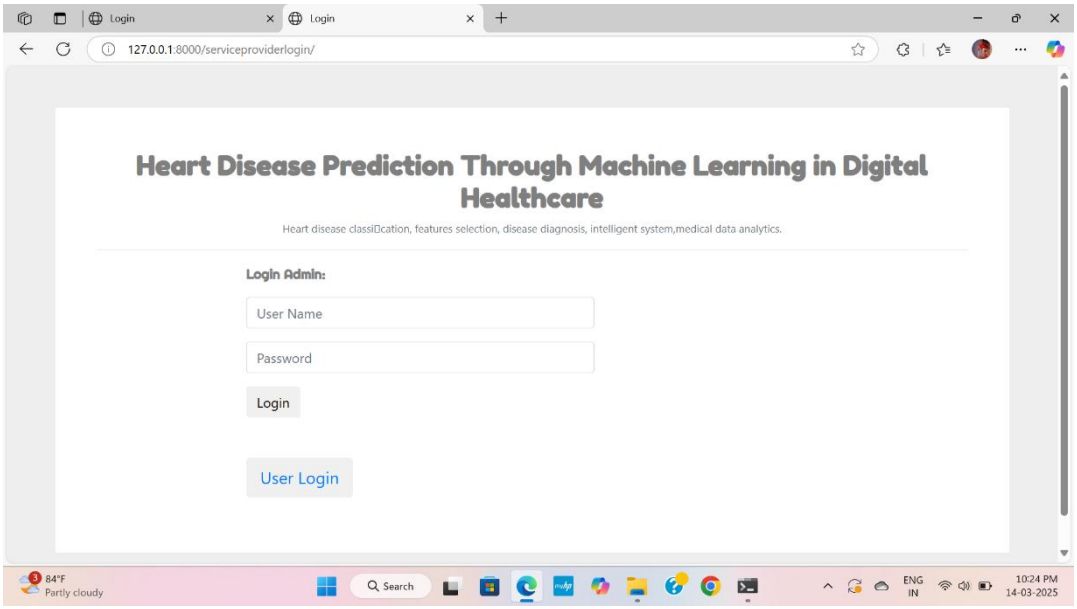
Screen 5.2.8 File Upload Page

Description: The above screen shows the Choose and uploading of file and we get the patient details.



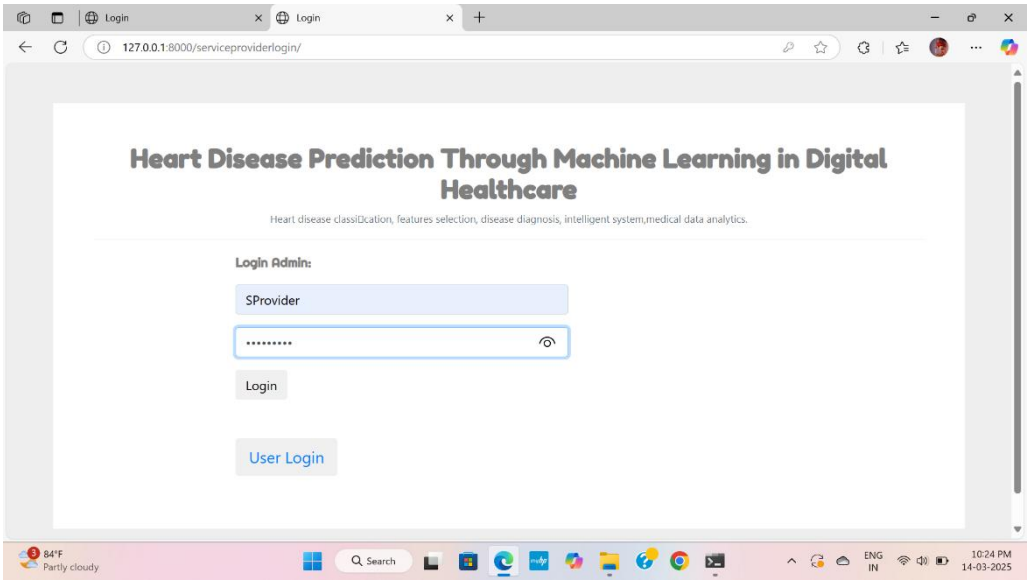
Screen 5.2.9 Patient Details page

Description: The above page shows the patient details.



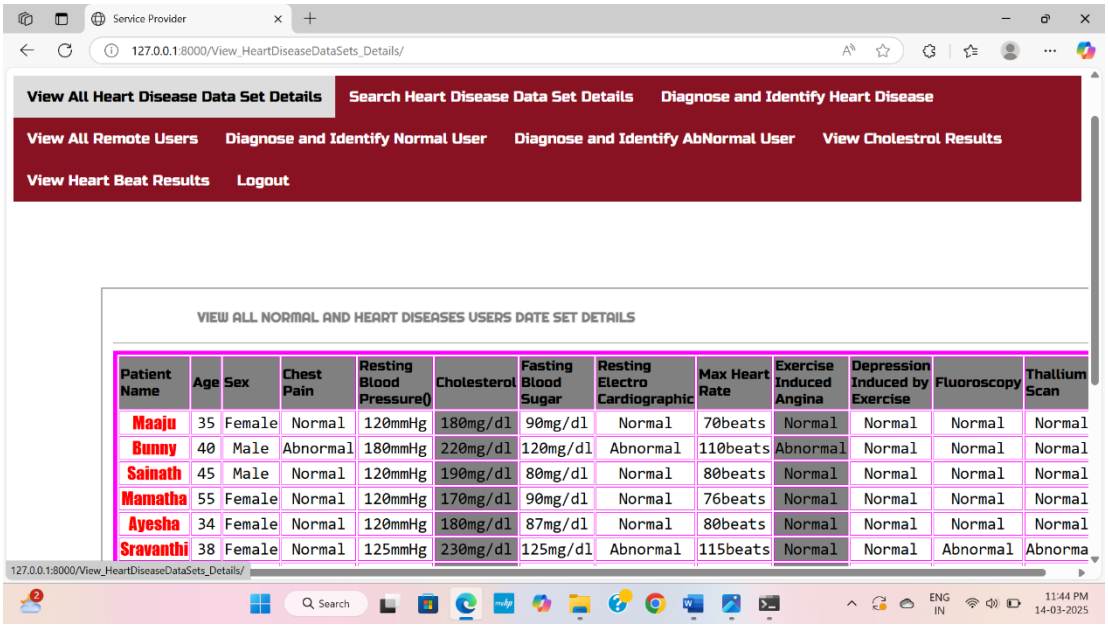
Screen 5.2.10 Admin Page

Description: The above page shows the admin Page of the User.



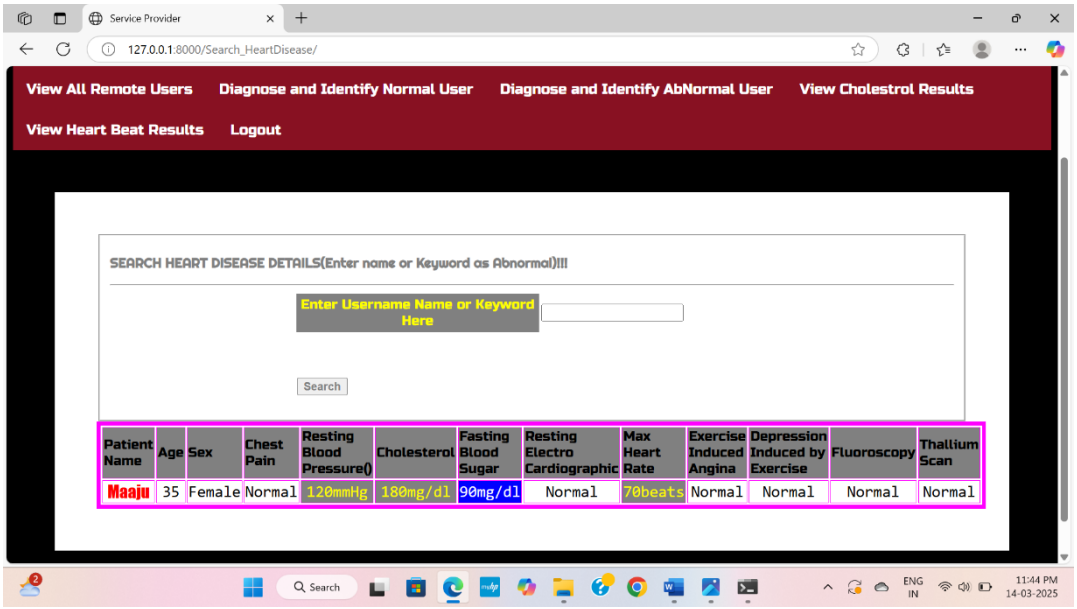
Screen 5.2.11 Admin Details Page

Description: The page shows the Admin Details of the Administrator.



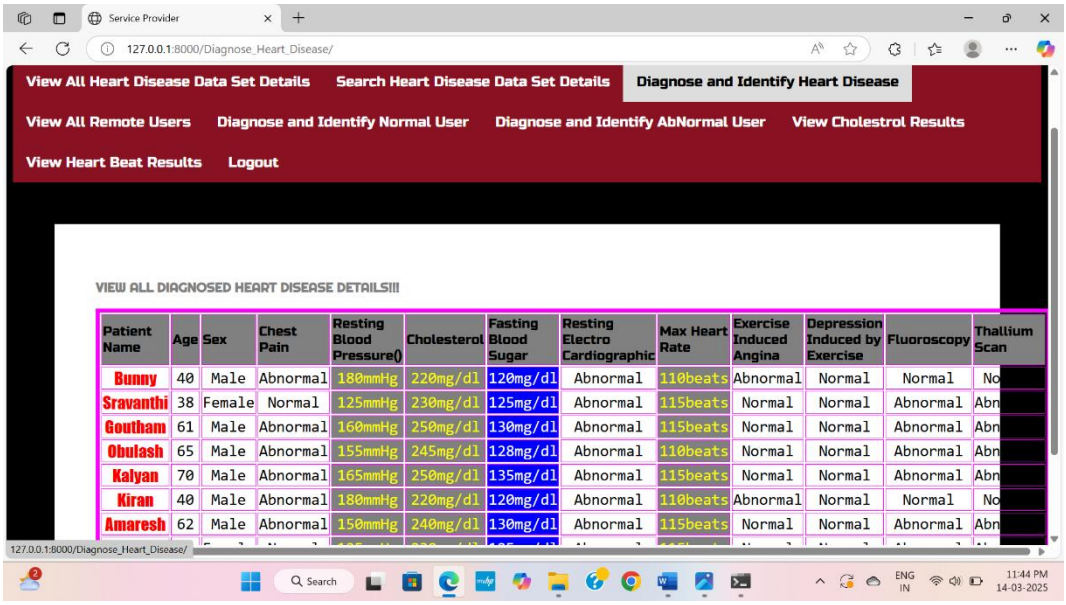
Screen 5.2.12 Data set Details Page

Description: The above page shows the Patient Details.



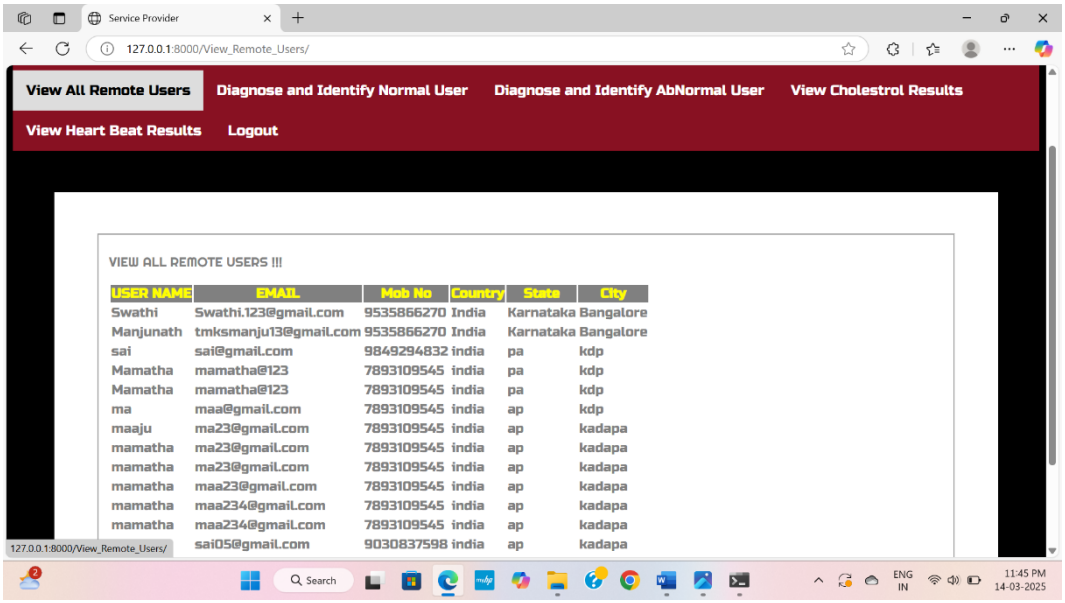
Screen 5.2.13 Searched Patient Details Page

Description: The Page shows the search a particular patient Details.



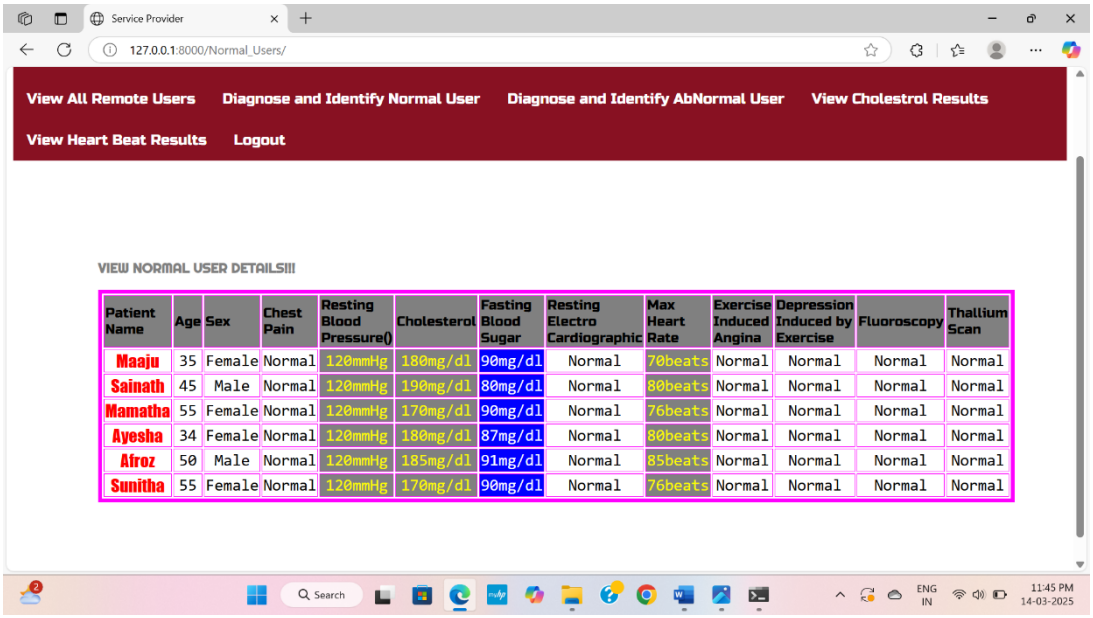
Screen 5.2.14 Patient Details Page

Description: The above pages shows the Diagnose and Identified Heart Disease Patient Details.



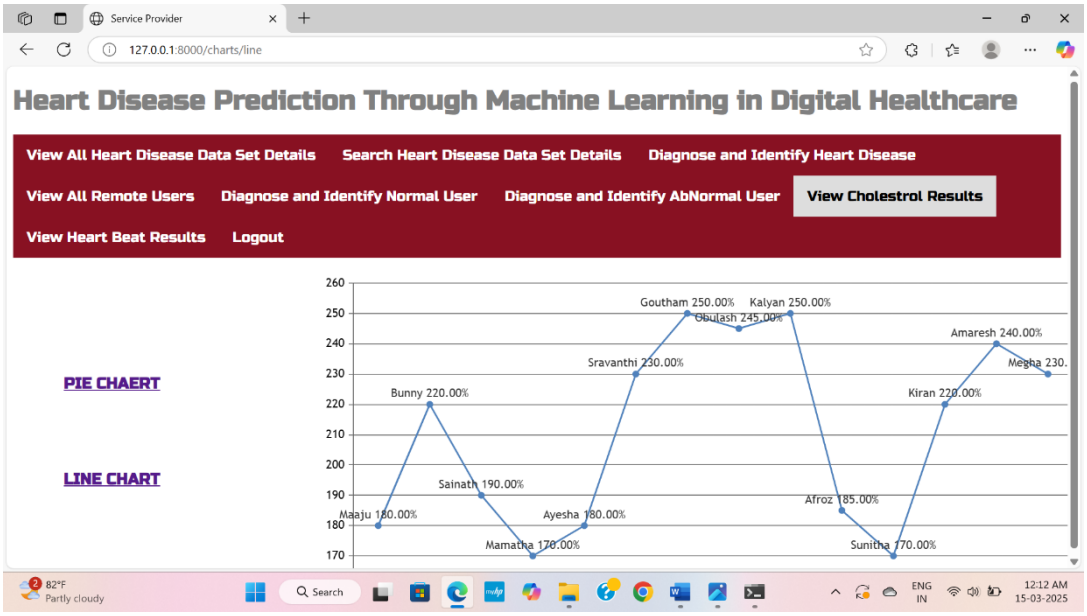
Screen 5.2.15 Remote Users Page

Description: The above page shows the View all remote users in the admin Page.



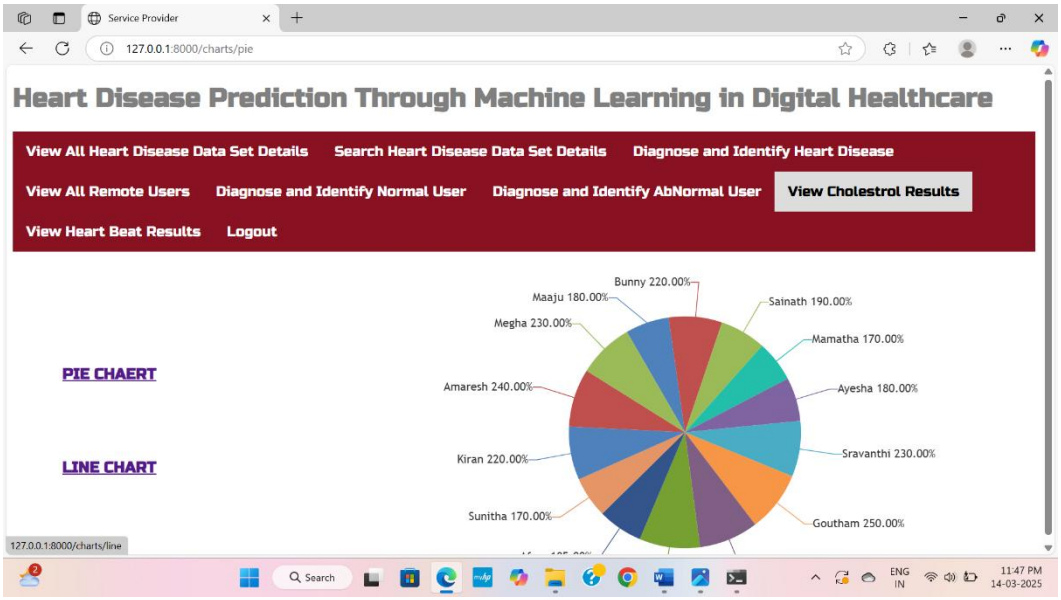
Screen 5.2.16 Patient Details Page

Description: The above page shows the Diagnose and Identified Normal Patient Details.



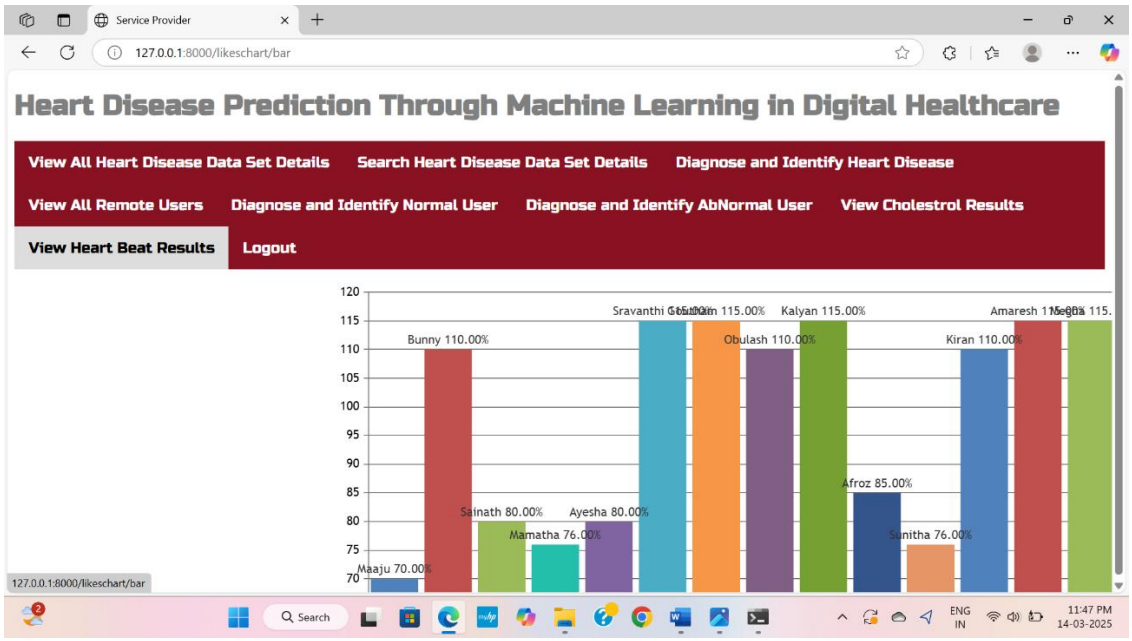
Screen 5.2.17 Line Chart page

Description: The above page shows the Cholestrol results in Line Chart.



Screen 5.2.18 Pie Chart Page

Description: The above page shows the Cholestrol results in Pie chart.



Screen 5.2.19 Bar chart Results Page

Description: The above page shows the Heart Beat Results in the Bar chart.

CONCLUSION

Chapter 6

CONCLUSION

A machine learning-based heart disease diagnosis system was developed using classifiers such as LR, K-NN, ANN, SVM, NB, and DT. The system also proposed a novel feature selection algorithm, FCMIM, which was compared with four standard algorithms: Relief, MRMR, LASSO, and LLBFS. When tested on the Cleveland heart disease dataset, the FCMIM algorithm outperformed the standard algorithms in feature selection. Additionally, Logistic Regression with Relief, LASSO, FCMIM, and LLBFS had the best processing time. The results identified Thallium Scan type, chest pain, and Exercise-induced Angina as important features, while Fasting blood sugar (FBS) was found to be an unsuitable feature for heart disease diagnosis. Overall, the proposed system and feature selection algorithm demonstrated improved performance and accuracy in heart disease diagnosis, highlighting the potential benefits of using machine learning algorithms in critical disease diagnosis.

6.1 Future Enhancement

To further enhance the machine learning-based heart disease diagnosis system, several potential future developments can be explored. Integrating additional heart disease datasets can validate the system's performance and accuracy. Investigating other machine learning algorithms, such as Random Forest, Gradient Boosting, or Ensemble methods, can also be beneficial. Furthermore, developing and incorporating additional relevant features, like medical history, lifestyle factors, or genetic information, can improve the system's effectiveness. Implementing interpretability techniques, such as feature importance or partial dependence plots, can provide valuable insights into the decision-making process. Other potential enhancements include real-time data processing from sources like wearable devices or electronic health records, clinical validation with medical professionals, developing a user-friendly interface, and implementing continuous learning mechanisms to adapt to changing patterns in heart disease diagnosis.

BIBLOGRAPHY

BIBLIOGRAPHY

K. Graves, Ceh: Official certified ethical hacker review guide: Exam 312-50. John Wiley & Sons, 2007.

R. Christopher, "Port scanning techniques and the defense against them," SANS Institute, 2001.

M. Baykara, R. Das¸, and I. Karado gan, "Bilgi g uvenil gi sistemlerinde kullanilan arac¸larin incelenmesi," in 1st International Symposium on Digital Forensics and Security (ISDFS13), 2013, pp. 231–239.

RashmiT V. "Predicting the System Failures Using Machine Learning Algorithms".International Journal of Advanced Scientific Innovation, vol. 1, no. 1, Dec. 2020, doi:10.5281/zenodo.4641686.

S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo, "Surveillance detection in high bandwidth environments," in DARPA Information Survivability Conference and Exposition, 2003. Proceedings, vol. 1. IEEE, 2003, pp. 130–138.

K. Ibrahimi and M. Ouaddane, "Management of intrusion detection systems based-kdd99: Analysis with lda and pca," in Wireless Networks and Mobile Communications (WINCOM), 2017 International Conference on. IEEE, 2017, pp. 1–6.

Girish L, Rao SKN (2020) "Quantifying sensitivity and performance degradation of virtual machines using machine learning.",Journal of Computational and Theoretical Nanoscience, Volume 17, Numbers 9-10, September/October 2020, pp.4055-4060(6) https://doi.org/10.1166/jctn.2020.9019

L. Sun, T. Anthony, H. Z. Xia, J. Chen, X. Huang, and Y. Zhang, "Detection and classification of malicious patterns in network traffic using benford's law," in Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017. IEEE, 2017, pp. 864–872.

S. M. Almansob and S. S. Lomte, "Addressing challenges for intrusion detection system using naive bayes and pca algorithm," in Convergence in Technology (I2CT), 2017 2nd International Conference for. IEEE, 2017, pp. 565–568

APPENDIX – A

- **URL LISTING**

www.google.co.in

www.python.org

www.w3schools.com

www.pythontutorial.com

- **REFERENCE BOOKS**

1. Python Crash Course 2nd Edition - this is a basic level book for beginners.
2. Learning python 5th Edition - this book is a practical learning book for basic to advanced level.
3. Python Cookbook - this book for advanced programmer interested in learning about modern python development tools.
4. Automating Boring Stuff With Python - In this book you will learn to write programs in python.
5. Head First Python - this book covered the fundamental of python.
6. Think Python - the basics of programming concepts and cover advanced topics like data structure and object-oriented design.

APPENDIX – B

- **GLOSSARY**

➢ GUI	:	Graphical User Interface
➢ UML	:	Unified Modeling Language
➢ API	:	Application Programming Interface
➢ HTML	:	Hyper Text Markup Language
➢ URL	:	Uniform Resource Locator
➢ ODBC	:	Open Database Connectivity

APPENDIX – C

List of Figures

List of Tables

List of Screens

APPENDIX – D

• Coding

```
from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime



# Create your views here.
from Remote_User.models
import
heart_disease_model,ClientRegister_Model,review_Model,recommend_Model



def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "SProvider" and password =="SProvider":
            return redirect('View_Remote_Users')

    return render(request,'SProvider/serviceproviderlogin.html')



def viewtreandingquestions(request,chart_type):
    dd = {}
    pos,neu,neg =0,0,0
    poss=None
    topic                                                                    =
heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order
_by('-dcount')
    for t in topic:
        topics=t['ratings']
```

```
pos_count=heart_disease_model.objects.filter(topics=topics).values('names').annotate
(topiccount=Count('ratings'))
        poss=pos_count
        for pp in pos_count:
            senti= pp['names']
            if senti == 'positive':
                pos= pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics]=[pos,neg,neu]
    return
render(request,'SProvider/viewtreandingquestions.html',{'object':topic,'dd':dd,'chart_t
ype':chart_type})

def Search_HeartDisease(request): # Search
    if request.method == "POST":
        kword = request.POST.get('keyword')
        obj = heart_disease_model.objects.all().filter(Q(chest_pain__contains=kword) |
Q(names__contains=kword) | Q(resting_electro_cardiographic__contains=kword)|
Q(exercise_induced_angina__contains=kword)|
Q(depression_induced_by_exercise__contains=kword)|
Q(fluoroscopy__contains=kword)| Q(thallium_scan__contains=kword))
        return render(request, 'SProvider/Search_HeartDisease.html', {'objs': obj})
    return render(request, 'SProvider/Search_HeartDisease.html')

def Diagnose_Heart_Disease(request): # Search

    bp=120
    cholestrol=200
    hrate_high=100
    hrate_low=60
    sugar=100

    obj = heart_disease_model.objects.all().filter(Q(resting_bp__gt=bp)|
Q(serum_cholesterol__gt=cholestrol)|Q(max_heart_rate__gt=hrate_high)|Q(max_hea
rt_rate__lt=hrate_low) | Q(fasting_blood_sugar__gt=sugar))
    return render(request, 'SProvider/Diagnose_Heart_Disease.html', {'objs': obj})


def Normal_Users(request): # Positive
```

```
    bp = 140
    cholestrol = 200
    hrate_high = 100
    hrate_low = 60
    sugar = 100
    obj = heart_disease_model.objects.all().filter(
        Q(resting_bp__lt=bp),
Q(serum_cholesterol__lt=cholestrol),Q(max_heart_rate__lt=hrate_high),Q(max_hear
t_rate__gt=hrate_low),Q(fasting_blood_sugar__lt=sugar))
    return render(request, 'SProvider/Normal_Users.html', {'objs': obj})

def Abnormal_Users(request):
    kword='Abnormal'
    obj = heart_disease_model.objects.all().filter(
        Q(chest_pain__contains=kword) |
Q(resting_electro_cardiographic__contains=kword)|
Q(exercise_induced_angina__contains=kword)|Q(depression_induced_by_exercise__
contains=kword),Q(
            fluoroscopy__contains=kword), Q(thallium_scan__contains=kword))
    return render(request, 'SProvider/Abnormal_Users.html', {'objs': obj})



def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def ViewTrendings(request):
    topic =
heart_disease_model.objects.values('topics').annotate(dcount=Count('topics')).order_b
y('-dcount')
    return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})

def negativechart(request,chart_type):
    dd = {}
    pos, neu, neg = 0, 0, 0
    poss = None
```

```
    topic = heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')
    for t in topic:
        topics = t['ratings']
        pos_count = heart_disease_model.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))
        poss = pos_count
        for pp in pos_count:
            senti = pp['names']
            if senti == 'positive':
                pos = pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics] = [pos, neg, neu]
    return render(request,'SProvider/negativechart.html',{'object':topic,'dd':dd,'chart_type':chart_type})


def charts(request,chart_type):
    chart1 = heart_disease_model.objects.values('names').annotate(dcount=Avg('serum_cholesterol'))
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def View_HeartDiseaseDataSets_Details(request):
    obj = heart_disease_model.objects.all()
    return render(request, 'SProvider/View_HeartDiseaseDataSets_Details.html', {'list_objects': obj})

def likeschart(request,like_chart):
```

```
    charts = heart_disease_model.objects.values('names').annotate(dcount=Avg('max_heart_rate'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})
```

```
from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404
import datetime
import openpyxl


# Create your views here.
from Remote_User.models import review_Model,ClientRegister_Model,heart_disease_model,recommend_Model


def login(request):


    if request.method == "POST" and 'submit1' in request.POST:

        username = request.POST.get('username')
        password = request.POST.get('password')
        try:

            enter = ClientRegister_Model.objects.get(username=username, password=password)
            request.session["userid"] = enter.id
            return redirect('Add_DataSet_Details')
        except:
            pass
```

```
    return render(request,'RUser/login.html')

def Add_DataSet_Details(request):
    if "GET" == request.method:
        return render(request, 'RUser/Add_DataSet_Details.html', {})
    else:
        excel_file = request.FILES["excel_file"]

        # you may put validations here to check extension or file size

        wb = openpyxl.load_workbook(excel_file)

        # getting all sheets
        sheets = wb.sheetnames
        print(sheets)

        # getting a particular sheet
        worksheet = wb["Sheet1"]
        print(worksheet)

        # getting active sheet
        active_sheet = wb.active
        print(active_sheet)

        # reading a cell
        print(worksheet["A1"].value)

        excel_data = list()
        # iterating over the rows and
        # getting value from each cell in row
        for row in worksheet.iter_rows():
            row_data = list()
            for cell in row:
                row_data.append(str(cell.value))
                print(cell.value)
```

```
            excel_data.append(row_data)

            heart_disease_model.objects.all().delete()

        for r in range(1, active_sheet.max_row+1):
            heart_disease_model.objects.create(
            names=active_sheet.cell(r, 1).value,
                age=active_sheet.cell(r, 2).value,
            sex=active_sheet.cell(r, 3).value,
            chest_pain=active_sheet.cell(r, 4).value,
            resting_bp=active_sheet.cell(r, 5).value,
            serum_cholesterol=active_sheet.cell(r, 6).value,
            fasting_blood_sugar=active_sheet.cell(r, 7).value,
            resting_electro_cardiographic=active_sheet.cell(r, 8).value,
            max_heart_rate=active_sheet.cell(r, 9).value,
            exercise_induced_angina=active_sheet.cell(r, 10).value,
            depression_induced_by_exercise=active_sheet.cell(r, 11).value,
            fluoroscopy=active_sheet.cell(r, 12).value,
            thallium_scan=active_sheet.cell(r, 13).value

            )

        return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data": excel_data})


def Register1(request):

    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
        state = request.POST.get('state')
        city = request.POST.get('city')
```

```
        ClientRegister_Model.objects.create(username=username, email=email,
password=password, phoneno=phoneno,
                                            country=country, state=state, city=city)

        return render(request, 'RUser/Register1.html')
    else:

        return render(request,'RUser/Register1.html')



def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request,'RUser/ViewYourProfile.html',{'object':obj})



def Search_Heart_Disease(request):

    if request.method == "POST":
        kword = request.POST.get('keyword')
        obj  =  heart_disease_model.objects.all().filter(Q(chest_pain__contains=kword)  |
Q(names__contains=kword) | Q(
            resting_electro_cardiographic__contains=kword)                              |
Q(exercise_induced_angina__contains=kword) | Q(
            depression_induced_by_exercise__contains=kword)                             |
Q(fluoroscopy__contains=kword) | Q(
            thallium_scan__contains=kword))
        return render(request, 'RUser/Search_Heart_Disease.html',{'objs': obj})
    return render(request, 'RUser/Search_Heart_Disease.html')



def ratings(request,pk):
    vott1, vott, neg = 0, 0, 0
    objs = heart_disease_model.objects.get(id=pk)
    unid = objs.id
    vot_count = heart_disease_model.objects.all().filter(id=unid)
```

```
    for t in vot_count:
        vott = t.ratings
        vott1 = vott + 1
        obj = get_object_or_404(heart_disease_model, id=unid)
        obj.ratings = vott1
        obj.save(update_fields=["ratings"])
        return redirect('Add_DataSet_Details')

    return render(request,'RUser/ratings.html',{'objs':vott1})


from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404
import datetime
import openpyxl


# Create your views here.
from Remote_User.models import review_Model,ClientRegister_Model,heart_disease_model,recommend_Model


def login(request):


    if request.method == "POST" and 'submit1' in request.POST:

        username = request.POST.get('username')
        password = request.POST.get('password')
        try:

            enter = ClientRegister_Model.objects.get(username=username, password=password)
            request.session["userid"] = enter.id
            return redirect('Add_DataSet_Details')
```

```
        except:
            pass

    return render(request,'RUser/login.html')

def Add_DataSet_Details(request):
    if "GET" == request.method:
        return render(request, 'RUser/Add_DataSet_Details.html', {})
    else:
        excel_file = request.FILES["excel_file"]

        # you may put validations here to check extension or file size

        wb = openpyxl.load_workbook(excel_file)

        # getting all sheets
        sheets = wb.sheetnames
        print(sheets)

        # getting a particular sheet
        worksheet = wb["Sheet1"]
        print(worksheet)

        # getting active sheet
        active_sheet = wb.active
        print(active_sheet)

        # reading a cell
        print(worksheet["A1"].value)

        excel_data = list()
        # iterating over the rows and
        # getting value from each cell in row
        for row in worksheet.iter_rows():
            row_data = list()
            for cell in row:
```

```
            row_data.append(str(cell.value))
            print(cell.value)
        excel_data.append(row_data)

        heart_disease_model.objects.all().delete()

    for r in range(1, active_sheet.max_row+1):
        heart_disease_model.objects.create(
        names=active_sheet.cell(r, 1).value,
            age=active_sheet.cell(r, 2).value,
        sex=active_sheet.cell(r, 3).value,
        chest_pain=active_sheet.cell(r, 4).value,
        resting_bp=active_sheet.cell(r, 5).value,
        serum_cholesterol=active_sheet.cell(r, 6).value,
        fasting_blood_sugar=active_sheet.cell(r, 7).value,
        resting_electro_cardiographic=active_sheet.cell(r, 8).value,
        max_heart_rate=active_sheet.cell(r, 9).value,
        exercise_induced_angina=active_sheet.cell(r, 10).value,
        depression_induced_by_exercise=active_sheet.cell(r, 11).value,
        fluoroscopy=active_sheet.cell(r, 12).value,
        thallium_scan=active_sheet.cell(r, 13).value

        )

    return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data": excel_data})


def Register1(request):

    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
```

```
        state = request.POST.get('state')
        city = request.POST.get('city')
        ClientRegister_Model.objects.create(username=username, email=email,
password=password, phoneno=phoneno,
                                            country=country, state=state, city=city)

        return render(request, 'RUser/Register1.html')
    else:

        return render(request,'RUser/Register1.html')



def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request,'RUser/ViewYourProfile.html',{'object':obj})



def Search_Heart_Disease(request):

    if request.method == "POST":
        kword = request.POST.get('keyword')
        obj  =  heart_disease_model.objects.all().filter(Q(chest_pain__contains=kword)  |
Q(names__contains=kword) | Q(
            resting_electro_cardiographic__contains=kword)                              |
Q(exercise_induced_angina__contains=kword) | Q(
            depression_induced_by_exercise__contains=kword)                             |
Q(fluoroscopy__contains=kword) | Q(
            thallium_scan__contains=kword))
        return render(request, 'RUser/Search_Heart_Disease.html',{'objs': obj})
    return render(request, 'RUser/Search_Heart_Disease.html')



def ratings(request,pk):
    vott1, vott, neg = 0, 0, 0
    objs = heart_disease_model.objects.get(id=pk)
```

```
    unid = objs.id
    vot_count = heart_disease_model.objects.all().filter(id=unid)
    for t in vot_count:
        vott = t.ratings
        vott1 = vott + 1
        obj = get_object_or_404(heart_disease_model, id=unid)
        obj.ratings = vott1
        obj.save(update_fields=["ratings"])
        return redirect('Add_DataSet_Details')

    return render(request,'RUser/ratings.html',{'objs':vott1})



from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime


# Create your views here.
from Remote_User.models import heart_disease_model,ClientRegister_Model,review_Model,recommend_Model


def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "SProvider" and password =="SProvider":
            return redirect('View_Remote_Users')

    return render(request,'SProvider/serviceproviderlogin.html')
```

```
def viewtreandingquestions(request,chart_type):
    dd = {}
    pos,neu,neg =0,0,0
    poss=None
    topic = heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')
    for t in topic:
        topics=t['ratings']

pos_count=heart_disease_model.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))
        poss=pos_count
        for pp in pos_count:
            senti= pp['names']
            if senti == 'positive':
                pos= pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics]=[pos,neg,neu]
    return render(request,'SProvider/viewtreandingquestions.html',{'object':topic,'dd':dd,'chart_type':chart_type})

def Search_HeartDisease(request): # Search
    if request.method == "POST":
        kword = request.POST.get('keyword')
        obj = heart_disease_model.objects.all().filter(Q(chest_pain__contains=kword) | Q(names__contains=kword) | Q(resting_electro_cardiographic__contains=kword)| Q(exercise_induced_angina__contains=kword)| Q(depression_induced_by_exercise__contains=kword)| Q(fluoroscopy__contains=kword)| Q(thallium_scan__contains=kword))
        return render(request, 'SProvider/Search_HeartDisease.html', {'objs': obj})
    return render(request, 'SProvider/Search_HeartDisease.html')
```

```
def Diagnose_Heart_Disease(request): # Search

    bp=120
    cholestrol=200
    hrate_high=100
    hrate_low=60
    sugar=100

    obj = heart_disease_model.objects.all().filter(Q(resting_bp__gt=bp)|
Q(serum_cholesterol__gt=cholestrol)|Q(max_heart_rate__gt=hrate_high)|Q(max_hea
rt_rate__lt=hrate_low) | Q(fasting_blood_sugar__gt=sugar))
    return render(request, 'SProvider/Diagnose_Heart_Disease.html', {'objs': obj})



def Normal_Users(request): # Positive

    bp = 140
    cholestrol = 200
    hrate_high = 100
    hrate_low = 60
    sugar = 100
    obj = heart_disease_model.objects.all().filter(
        Q(resting_bp__lt=bp),
Q(serum_cholesterol__lt=cholestrol),Q(max_heart_rate__lt=hrate_high),Q(max_hear
t_rate__gt=hrate_low),Q(fasting_blood_sugar__lt=sugar))
    return render(request, 'SProvider/Normal_Users.html', {'objs': obj})

def Abnormal_Users(request):
    kword='Abnormal'
    obj = heart_disease_model.objects.all().filter(
        Q(chest_pain__contains=kword) |
Q(resting_electro_cardiographic__contains=kword)|
Q(exercise_induced_angina__contains=kword)|Q(depression_induced_by_exercise__
contains=kword),Q(
            fluoroscopy__contains=kword), Q(thallium_scan__contains=kword))
```

```
    return render(request, 'SProvider/Abnormal_Users.html', {'objs': obj})



def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def ViewTrendings(request):
    topic = heart_disease_model.objects.values('topics').annotate(dcount=Count('topics')).order_by('-dcount')
    return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})

def negativechart(request,chart_type):
    dd = {}
    pos, neu, neg = 0, 0, 0
    poss = None
    topic = heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')
    for t in topic:
        topics = t['ratings']
        pos_count = heart_disease_model.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))
        poss = pos_count
        for pp in pos_count:
            senti = pp['names']
            if senti == 'positive':
                pos = pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics] = [pos, neg, neu]
```

```
    return
render(request,'SProvider/negativechart.html',{'object':topic,'dd':dd,'chart_type':chart_
type})



def charts(request,chart_type):
    chart1 =
heart_disease_model.objects.values('names').annotate(dcount=Avg('serum_cholestero
l'))
    return render(request,"SProvider/charts.html", {'form':chart1,
'chart_type':chart_type})

def View_HeartDiseaseDataSets_Details(request):
    obj = heart_disease_model.objects.all()
    return render(request, 'SProvider/View_HeartDiseaseDataSets_Details.html',
{'list_objects': obj})

def likeschart(request,like_chart):
    charts =
heart_disease_model.objects.values('names').annotate(dcount=Avg('max_heart_rate'))
    return render(request,"SProvider/likeschart.html", {'form':charts,
'like_chart':like_chart})









from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime



# Create your views here.
```

```
from Remote_User.models import heart_disease_model,ClientRegister_Model,review_Model,recommend_Model


def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "SProvider" and password =="SProvider":
            return redirect('View_Remote_Users')

    return render(request,'SProvider/serviceproviderlogin.html')



def viewtreandingquestions(request,chart_type):
    dd = {}
    pos,neu,neg =0,0,0
    poss=None
    topic = heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')
    for t in topic:
        topics=t['ratings']

    pos_count=heart_disease_model.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))
        poss=pos_count
        for pp in pos_count:
            senti= pp['names']
            if senti == 'positive':
                pos= pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics]=[pos,neg,neu]
```

```
    return
render(request,'SProvider/viewtreandingquestions.html',{'object':topic,'dd':dd,'chart_t
ype':chart_type})

def Search_HeartDisease(request): # Search
    if request.method == "POST":
        kword = request.POST.get('keyword')
        obj = heart_disease_model.objects.all().filter(Q(chest_pain__contains=kword) |
Q(names__contains=kword) | Q(resting_electro_cardiographic__contains=kword)|
Q(exercise_induced_angina__contains=kword)|
Q(depression_induced_by_exercise__contains=kword)|
Q(fluoroscopy__contains=kword)| Q(thallium_scan__contains=kword))
        return render(request, 'SProvider/Search_HeartDisease.html', {'objs': obj})
    return render(request, 'SProvider/Search_HeartDisease.html')

def Diagnose_Heart_Disease(request): # Search

    bp=120
    cholestrol=200
    hrate_high=100
    hrate_low=60
    sugar=100

    obj = heart_disease_model.objects.all().filter(Q(resting_bp__gt=bp)|
Q(serum_cholesterol__gt=cholestrol)|Q(max_heart_rate__gt=hrate_high)|Q(max_hea
rt_rate__lt=hrate_low) | Q(fasting_blood_sugar__gt=sugar))
    return render(request, 'SProvider/Diagnose_Heart_Disease.html', {'objs': obj})


def Normal_Users(request): # Positive

    bp = 140
    cholestrol = 200
    hrate_high = 100
    hrate_low = 60
    sugar = 100
```

```
    obj = heart_disease_model.objects.all().filter(
        Q(resting_bp__lt=bp),
Q(serum_cholesterol__lt=cholestrol),Q(max_heart_rate__lt=hrate_high),Q(max_hear
t_rate__gt=hrate_low),Q(fasting_blood_sugar__lt=sugar))
    return render(request, 'SProvider/Normal_Users.html', {'objs': obj})

def Abnormal_Users(request):
    kword='Abnormal'
    obj = heart_disease_model.objects.all().filter(
        Q(chest_pain__contains=kword) |
Q(resting_electro_cardiographic__contains=kword)|
Q(exercise_induced_angina__contains=kword)|Q(depression_induced_by_exercise__
contains=kword),Q(
            fluoroscopy__contains=kword), Q(thallium_scan__contains=kword))
    return render(request, 'SProvider/Abnormal_Users.html', {'objs': obj})



def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def ViewTrendings(request):
    topic =
heart_disease_model.objects.values('topics').annotate(dcount=Count('topics')).order_b
y('-dcount')
    return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})

def negativechart(request,chart_type):
    dd = {}
    pos, neu, neg = 0, 0, 0
    poss = None
    topic =
heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order
_by('-dcount')
    for t in topic:
        topics = t['ratings']
```

```
        pos_count = heart_disease_model.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))
        poss = pos_count
        for pp in pos_count:
            senti = pp['names']
            if senti == 'positive':
                pos = pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics] = [pos, neg, neu]
    return render(request,'SProvider/negativechart.html',{'object':topic,'dd':dd,'chart_type':chart_type})


def charts(request,chart_type):
    chart1 = heart_disease_model.objects.values('names').annotate(dcount=Avg('serum_cholesterol'))
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def View_HeartDiseaseDataSets_Details(request):
    obj = heart_disease_model.objects.all()
    return render(request, 'SProvider/View_HeartDiseaseDataSets_Details.html', {'list_objects': obj})

def likeschart(request,like_chart):
    charts = heart_disease_model.objects.values('names').annotate(dcount=Avg('max_heart_rate'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})
```

```
from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime


# Create your views here.
from Remote_User.models import heart_disease_model,ClientRegister_Model,review_Model,recommend_Model


def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "SProvider" and password =="SProvider":
            return redirect('View_Remote_Users')

    return render(request,'SProvider/serviceproviderlogin.html')


def viewtreandingquestions(request,chart_type):
    dd = {}
    pos,neu,neg =0,0,0
    poss=None
    topic = heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')
    for t in topic:
        topics=t['ratings']
```

```
pos_count=heart_disease_model.objects.filter(topics=topics).values('names').annotate
(topiccount=Count('ratings'))
        poss=pos_count
        for pp in pos_count:
            senti= pp['names']
            if senti == 'positive':
                pos= pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics]=[pos,neg,neu]
    return
render(request,'SProvider/viewtreandingquestions.html',{'object':topic,'dd':dd,'chart_t
ype':chart_type})

def Search_HeartDisease(request): # Search
    if request.method == "POST":
        kword = request.POST.get('keyword')
        obj  =  heart_disease_model.objects.all().filter(Q(chest_pain__contains=kword)  |
Q(names__contains=kword)   |   Q(resting_electro_cardiographic__contains=kword)|
Q(exercise_induced_angina__contains=kword)|
Q(depression_induced_by_exercise__contains=kword)|
Q(fluoroscopy__contains=kword)| Q(thallium_scan__contains=kword))
        return render(request, 'SProvider/Search_HeartDisease.html', {'objs': obj})
    return render(request, 'SProvider/Search_HeartDisease.html')

def Diagnose_Heart_Disease(request): # Search

    bp=120
    cholestrol=200
    hrate_high=100
    hrate_low=60
    sugar=100
```

```
    obj = heart_disease_model.objects.all().filter(Q(resting_bp__gt=bp)|
Q(serum_cholesterol__gt=cholestrol)|Q(max_heart_rate__gt=hrate_high)|Q(max_hea
rt_rate__lt=hrate_low) | Q(fasting_blood_sugar__gt=sugar))
    return render(request, 'SProvider/Diagnose_Heart_Disease.html', {'objs': obj})


def Normal_Users(request): # Positive

    bp = 140
    cholestrol = 200
    hrate_high = 100
    hrate_low = 60
    sugar = 100
    obj = heart_disease_model.objects.all().filter(
        Q(resting_bp__lt=bp),
Q(serum_cholesterol__lt=cholestrol),Q(max_heart_rate__lt=hrate_high),Q(max_hear
t_rate__gt=hrate_low),Q(fasting_blood_sugar__lt=sugar))
    return render(request, 'SProvider/Normal_Users.html', {'objs': obj})

def Abnormal_Users(request):
    kword='Abnormal'
    obj = heart_disease_model.objects.all().filter(
        Q(chest_pain__contains=kword) |
Q(resting_electro_cardiographic__contains=kword)|
Q(exercise_induced_angina__contains=kword)|Q(depression_induced_by_exercise__
contains=kword),Q(
            fluoroscopy__contains=kword), Q(thallium_scan__contains=kword))
    return render(request, 'SProvider/Abnormal_Users.html', {'objs': obj})


def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def ViewTrendings(request):
```

```
    topic = heart_disease_model.objects.values('topics').annotate(dcount=Count('topics')).order_by('-dcount')
    return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})

def negativechart(request,chart_type):
    dd = {}
    pos, neu, neg = 0, 0, 0
    poss = None
    topic = heart_disease_model.objects.values('ratings').annotate(dcount=Count('ratings')).order_by('-dcount')
    for t in topic:
        topics = t['ratings']
        pos_count = heart_disease_model.objects.filter(topics=topics).values('names').annotate(topiccount=Count('ratings'))
        poss = pos_count
        for pp in pos_count:
            senti = pp['names']
            if senti == 'positive':
                pos = pp['topiccount']
            elif senti == 'negative':
                neg = pp['topiccount']
            elif senti == 'nutral':
                neu = pp['topiccount']
        dd[topics] = [pos, neg, neu]
    return render(request,'SProvider/negativechart.html',{'object':topic,'dd':dd,'chart_type':chart_type})


def charts(request,chart_type):
    chart1 = heart_disease_model.objects.values('names').annotate(dcount=Avg('serum_cholesterol'))
```

```
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def View_HeartDiseaseDataSets_Details(request):
    obj = heart_disease_model.objects.all()
    return render(request, 'SProvider/View_HeartDiseaseDataSets_Details.html', {'list_objects': obj})

def likeschart(request,like_chart):
    charts = heart_disease_model.objects.values('names').annotate(dcount=Avg('max_heart_rate'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})
```