**DISEASE PREDICTION AND MEDICAL CHECK-UP USING DEEP LEARNING**

### A PROJECT REPORT

***Submitted by***

|  |  |
| --- | --- |
| **SAVARAM BALAJAYANTH** | **(211418104237)** |
| **VAMSI KRISHNA. J** | **(211418104301)** |
| **SARAVANAN. B** | **(211418104329)** |

***in partial fulfillment for the award of the degree of***

## BACHELOR OF ENGINEERING

***in***

### COMPUTER SCIENCE AND ENGINEERING



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

### MAY 2022

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## BONAFIDE CERTIFICATE

Certified that this project report **“DISEASE PREDICTION AND MEDICAL CHECK-UP USING DEEP LEARNING”** is the bonafide work of “**SAVARAM BALAJAYANTH (211418104237), VAMSI KRISHNA. J (211418104301), SARAVANAN. B(211418104329)”** who carried out the project work under my supervision.

### SIGNATURE SIGNATURE

### Dr.S.MURUGAVALLIM.E.,Ph.D., Mr.N.PUGHAZWNDHI,M.E.,Ph.D.,

**HEAD OFTHEDEPARTMENT PROFESSOR**

DEPARTMENTOFCSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERINGCOLLEGE PANIMALAR ENGINEERINGCOLLEGE

NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600123. CHENNAI-600123.

Certified that the above mentioned students were examined in the End Semester project viva-voceheldon .

### INTERNALEXAMINER EXTERNALEXAMINER

**DECLARATION BY THE STUDENT**

We **SAVARAM BALAJAYANTH (211418104237), VAMSI KRISHNA. J (211418104301), SARAVANAN. B (211418104329),**hereby declare that this project report titled **“DISEASE PREDICTION AND MEDICAL CHECK-UP USING DEEP LEARNING”,** under the guidance of **Dr.N.PUGHAZENDHI, M.E.,Ph.D**., is the original work done by us and we have not plagiarized orsubmittedto any other degree in any university by us.

## ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr. C. SAKTHI KUMAR, M.E., Ph.D**. and **Dr. SARANYASREE SAKTHI**

**KUMAR B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. S.MURUGAVALLI, M.E., Ph.D.,**

for the support extended throughout the project.

We would like to thank my **Project Coordinator** and **Project Guide Dr. N. PUGHAZENDI M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

SAVARAM BALAJAYANTH

VAMSI KRISHNA. J

SARAVANAN. B

**ABSTRACT**

Early diseases prediction plays an important role in improving healthcare quality and can help individuals avoid dangerous health situations before it is too late.The deep learning field has shown greater potential in applications such as disease prediction and drug response prediction. Deep learning techniques have continuously improved the accuracy of medical disease prediction as well as helped shape overall healthcare. Input is obtained in the form of numeric values for diseases such as cancer, diabetes, and cardiovascular conditions. To predict diseases such as Pneumonia, malaria, or kidney diseases, an image is used as an input. Output results are acquired instantly in real-time in the form of a probable disease, its causes, and the aftermath. We will be using Deep Learning algorithms, mainly CNN and DNN, and Image Processing.

**TABLE OF CONTENT**

| **CHAPTER** | **TITLE** | **PAGE NO** |
| --- | --- | --- |
|  | ABSTRACT | **v** |
|  | LIST OF FIGURES | **viii** |
|  | LIST OF ACRONYMS AND ABBREVIATIONS | **ix** |
| **1** | **INTRODUCTION**   * 1. INTRODUCTION   2. IDEOLOGY   3. CNN ON MEDICAL IMAGE CLASSIFICATION | **1**  **2**  **3**  **4** |
| **2** | **LITERATURE SURVEY** | **6** |
| **3** | **SYSTEM ANALYSIS**  3.1 EXISTING SYSTEM  3.2 PROPOSED SYSTEM  3.3 REQUIREMENT SPECIFICATION  3.4 LANGUAGE SPECIFICATION | **13**  **14**  **14**  **15**  **15** |
| **4** | **SYSTEM DESIGN**  4.1 SYSTEM ARCHITECTURE  4.2 USE-CASE DIAGRAM  4.3 ACTIVITY DIAGRAM  4.4 SEQUENCE DIAGRAM  4.5 DATA FLOW DIAGRAM 0  4.6 DATA FLOW DIAGRAM 1  4.7 ER DIAGRAM | **20**  **21**  **21**  **22**  **23**  **24**  **25**  **25** |
| **5** | **MODULE DESCRIPTION**  5.1 GETTING THE INPUT  5.2 CANCER, DIABETES HEART DISEASE PREDICTION  5.3CNN FOR IMAGE CLASSIFICATION  5.4 LOGISTIC REGRESSION FOR NUMERIC  5.5 END TO END DIAGNOSIS | **27**  **28**  **29**  **30**  **31**  **32** |
| **6** | **IMPLEMENTATION** | **34** |
| **7** | **TESTING**  7.1 UNIT TESTING | **40**  **41** |
| **8** | **CONCLUSION** | **42** |
| **9** | **APPENDIX 1** | **45** |
| **10** | **APPENDIX 2** | **59** |
| **11** | **REFERENCES** | **65** |

**LIST OF FIGURES**

| **FIG. NO** | **TITLE** | **PAGE NO** |
| --- | --- | --- |
| 4.1 | SYSTEM ARCHITECTURE | 21 |
| 4.2 | USE-CASE DIAGRAM | 22 |
| 4.3 | ACTIVITY DIAGRAM | 23 |
| 4.4 | SEQUENCE DIAGRAM | 24 |
| 4.5 | DATA FLOW DIAGRAM 0 | 24 |
| 4.6 | DATA FLOW DIAGRAM 1 | 25 |
| 4.7 | ER DIAGRAM | 26 |
| 5.1 | INPUT VALUES FOR ALL DISEASES | 28 |
| 5.2 | CNN ARCHITECTURE | 31 |
| 5.3 | FLOW OF END - END DIAGNOSIS | 32 |
| 6.1 | CONVOLUTION METHODOLOGY | 38 |
| 6.2 | MAXPOOLING | 38 |
| 6.3 | CONVERTING MATRIX INTO LINEAR ARRAY FORM | 39 |
| 9.1 | PARASITIC MALARIA PREDICTED | 59 |
| 9.2 | NORMAL LUNGS PREDICTED | 59 |
| 9.3 | DIABETES - INPUT | 60 |
| 9.4 | DIABETIC NOT PREDICTED | 61 |
| 9.5 | HEART DISEASE INPUT | 61 |
| 9.6 | HEART DISEASE PREDICTED | 62 |
| 9.7 | MALARIA IMAGE INPUT | 63 |
| 9.8 | PNEUMONIA IMAGE INPUT | 63 |

**LIST OF ACRONYMS AND ABBREVIATIONS**

**OCT -** Optical Coherence Tomography

**CNN -** Convolutional Neural Network

**SVM -** Support Vector Machine

**KNN -** K Nearest Neighbour

**INTRODUCTION**

**CHAPTER 1**

**INTRODUCTION**

**1.1 Introduction**

Accurately predicting diseases plays a significant role in public health, especially at the early stage which allows patients to take prevention treatments in time. With the growing volume and availability of electronic health records (EHRs), predictive modeling tasks for disease progression and analysis have obtained increasing interest from researchers. The EHR data are temporally sequenced by patient visits with each visit represented as a set of high dimensional clinical events. Mining EHRs is especially challenging compared to standard data mining tasks, due to their noisy, irregular, and heterogeneous nature. A conventional approach of disease prediction is the one-size-fits-all model. That is, using all available training data to build a global model, and then with this model, predicting the risk of diseases for each patient. The benefit of applying a one-size-fits-all model is that it captures the overall information of the entire training population. However, patients may have different phenotypes, different medical conditions, etc. Using a global model may miss some specific information that is important for individual patients. Thus, building a targeted, patient-specific model for each individual patient is urgent and important for personalized medicine. Recent studies show that personalized models can improve predictive performance over global models. A general framework for personalized prediction contains two stages: (1) measuring the similarity among patients, and (2) building a separate model for each patient using his/her similar cohorts. This framework is motivated by the working process of human doctors, i.e., after reviewing or recalling the diagnosed patients with similar diseases or symptoms, the doctors then carefully make decisions. If doctors can find similar patients, the probability of successfully curing this patient may improve a lot. Many similar learning methods have been proposed on healthcare datasets.

However, these models are developed for handcrafted vector representations such as demographics or average numerical values, without considering the temporal information from different visits. For the longitudinal EHR data, the number of patient visits varies largely, due to patients’ irregular visits and incomplete recordings. The aforementioned learning metrics cannot be directly applied to the longitudinal data, since the historical records of each patient do not naturally form a comparable vector. Therefore, one of the key challenges in measuring patient similarity is to derive an effective representation for each patient without the loss of his/her historical information. Recently, deep learning approaches have been widely adopted and rapidly developed in inpatient representation learning such as autoencoder, recurrent neural networks (RNNs), and convolution neural networks (CNNs). CNN has shown its superior ability in the task of measuring patient similarity. However, one drawback of the traditional CNN architecture is that it could not fully utilize the temporal and contextual information of EHRs for disease prediction. Consequently, simultaneously modeling the temporality and content of EHR data is more challenging.

**1.2 Ideology**

To tackle the aforementioned challenges and issues, in this paper, we aim to solve the following key problems in personalized prediction: how to build a model to accurately measure patients’ similarities from their historical records, and how to build an accurate personalized prediction model with the learned similarities. To achieve these goals, we first design a novel time-fusion CNN-based framework to account for the temporality across different time intervals. With the proposed framework, we can generate the vector representation for each patient. Based on the learned patient representations, a matching metric is then introduced to obtain a similar representation. Considering the practical meaning, we add a firm symmetric constraint to the framework structure. This similarity learning framework is end-to-end, which learns patient representations and pairwise similarity simultaneously. Since the similarity probability between a pair of patients indicates the risk level of the two patients developing the same disease, we use it as the score to rank the similarity among patients.

**1.3 CNN on medical image classification**

With the different CNN-based deep neural networks developed and achieved a significant result on ImageNet Challenger, which is the most significant image classification and segmentation challenge in the image analyzing field. The CNN-based deep neural system is widely used in the medical classification task. CNN is an excellent feature extractor, therefore utilizing it to classify medical images can avoid complicated and expensive feature engineering. Presented a customized CNN with shallow ConvLayer to classify image patches of lung disease. The authors also found that the system can be generalized to other medical image datasets. Moreover, in other research, it also found that CNN based system can be trained from a big chest X-ray (CXR) film dataset and state-of-art with high accuracy and sensitivity results on their dataset, like Stanford Normal Radiology Diagnostic Dataset containing more than 400,000 CXR and a new CXR database (ChestX-ray8), which consist of 108,948 frontal-view CXR. Moreover, using limited data makes it hard to train an adequate model.

Therefore, the transfer learning of CNN is widely used in medical image classification tasks. Germany et al. use InceptionV3 with ImageNet trained weight and transfer learning on a medical image dataset containing 108,312 optical coherence tomography (OCT) images. They got an average accuracy of 96.6%, with a sensitivity of 97.8% and a specificity of 97.4%. The authors also compared the results with six human experts. Most of the experts got high sensitivity but low specificity, while the CNN-based system got high values on both sensitivity and specificity. Moreover, on the average weight error measure, the CNN-based system exceeds two human experts. The authors also verified their system on a small pneumonia dataset, including about five thousand images, and achieved an average accuracy of 92.8%, with a sensitivity of 93.2% and a specificity of 90.1%. This system finally may help in accelerating diagnosis and referral of patients and therefore introduce early treatment, resulting in an increased cure rate. Moreover, Vianna also studied how to utilize transfer learning to build an X-ray image classification system that is the critical component of a computer-aided diagnosis system. The authors found a fine-tuned transfer learning system with data augmentation effectively alleviates the overfitting problem and yields a better result than two other models: training from scratch and a transfer learning model with only a retrained last classification layer.

# LITERATURE SURVEY

**CHAPTER 2**

**LITERATURE REVIEWS**

**TITLE 1:** A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis,The Lancet Digital Health Volume 1, Issue 6, October 2019

**AUTHORS:**

LivaFaesXiaoxuan Liu Pearse A Kean[1]

**DESCRIPTION:**

Diagnosis of disease using deep learning algorithms holds enormous potential. The accuracy of deep learning algorithms is equivalent to health-care professionals, while acknowledging that more studies considering the integration of such algorithms in real-world settings are needed. The more important finding around methodology and reporting means the credibility and path to impact of such diagnostic algorithms might be undermined by an excessive claim from a poorly designed or inadequately reported study. In this review, we have highlighted key issues of design and reporting that investigators should consider.

**TITLE 2 :**

Disease Prediction by Machine Learning Over Big Data From Healthcare Communities,

**AUTHORS:**

Min Chen, YixueHao, Kai Hwang, Lu Wang, Lin Wang[2]

**DESCRIPTION:**

The authors propose a new convolutional neural network (CNN)-based multimodal disease risk prediction algorithm using structured and unstructured data from the hospital. To the best of our knowledge, none of the existing work focused on both data types in the area of medical big data analytics. Compared with several typical prediction algorithms, the prediction accuracy of our proposed algorithm reaches 94.8% with a convergence speed, which is faster than that of the CNN-based unimodal disease risk prediction algorithm.

**TITLE 3 :**

Diagnosing of disease using machine learning,2021, Machine Learning and the Internet of Medical Things in Healthcare Pages 89-111

**AUTHORS:**

PushpaSingh,NarendraSingh,Krishna Kant Singh,Akansha Singh[3]

**DESCRIPTION:**

Disease diagnosis using machine-learning techniques like algorithms like SVM, KNN, Naive Bayes, and Decision tree can enhance the quickness of decision-making, and it can reduce the rate of false positives. The practical implementation of these algorithms is discussed with Python. These algorithms are used to diagnose various diseases like cancer, diabetes, epilepsy, heart attack, and other prominent diseases.

**TITLE 4 :**

An overview of deep learning in medical imaging focusing on MRI,May 2019,  ZeitschriftfürMedizinischePhysik Volume 29, Issue 2, Pages 102-127

**AUTHORS:**

Alexander SelvikvåglundervoldArvid  Lundervold[4]

**DESCRIPTION:**

Deep neural networks are now the state-of-the-art machine learning models across a variety of areas, from image analysis to natural language processing, and widely deployed in academia and industry. These developments have a huge potential for medical imaging technology, medical data analysis, medical diagnostics and healthcare in general, slowly being realized.

**TITLE 5 :**

Enhanced Deep Learning Assisted Convolutional Neural Network for Heart Disease Prediction on the Internet of Medical Things Platform,May 2020, IEEE Access ( Volume: 8)

**AUTHORS:**

Yuanyuan Pan; MinghuanFu;BiaoCheng;Xuefei Tao; Jing Guo[5]

**DESCRIPTION:**

The Enhanced Deep learning assisted Convolutional Neural Network (CNN) has been proposed to assist and improve patient prognostics of heart disease. The EDCNN model is focused on a deeper architecture which covers the multilayer perceptron's model with regularization learning approaches. The EDCNN system has been implemented on the Internet of Medical Things Platform (IoMT) for decision support systems which helps doctors to effectively diagnose heart patient's information in cloud platforms anywhere in the world.

**TITLE 6 :**

Designing Disease Prediction Model Using Machine Learning Approach, 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)

**AUTHORS:**

hirajDahiwade; GajananPatle; EktaaMeshram[6]

**DESCRIPTION:**

For disease prediction, we use K-Nearest Neighbor (KNN) and Convolutional neural network (CNN) machine learning algorithms for accurate prediction of disease. For disease prediction required disease symptoms dataset. In this general disease prediction the living habits of a person and checkup information are considered for the accurate prediction. The accuracy of general disease prediction by using CNN is 84.5% which is more than KNN algorithm. And the time and the memory requirement is also more in KNN than CNN.

**TITLE 7 :**

Healthcare Techniques Through Deep Learning:Issues, Challenges and Opportunities

**AUTHORS:**

Durr-e-maknoonNisarMeshrifAlruily Sultan H. Almotir Rashid Amin[7]

**DESCRIPTION:**

Tools and algorithms are the major parts of any system.In deep learning, tools are used to convert data into actionableinfo. The deep learning models are categorized according to supervised and unsupervised learning techniques. Due to these algorithms’ significant importance, medical devices should be built over time based on these models, such as the microscope, phonendoscope, and electrocardiogram, accommodating physicians’ perceptive capability limitations. In processing and forecasting from large datasets, deep learning is playing a critical role

**TITLE 8 :**

A Survey on Computer Vision for Assistive Medical Diagnosis From Faces,SEPTEMBER 2018, IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 22, NO. 5,

**AUTHORS:**

Jer´ omeThevenotˆ , Miguel Bordallo Lopez ´ , and AbdenourHadid[8]

**DESCRIPTION:**

a range of distant and non-invasive imaging solutions has been described, providing insights into the suitability of each one of the techniques in the assessment of different types of symptoms related to medical conditions. The findings show that utilizing computer vision methods, over 30 conditions can be preliminary diagnosed from the automatic detection of some of their symptoms

**TITLE 9 :**

Early-Stage Risk Prediction of Non-Communicable Disease UsingMachine Learning in Health CPS July 2021, IEEE Special Section On AI And IoT Convergence For Smart Health

**AUTHORS:**

RahataraFerdousi, M. Anwar Hossain, Abdulmotaleb El Saddik[9]

**DESCRIPTION:**

we propose a novel machine learning based health CPS framework that addresses the challenge of effectivelyprocessing the wearable IoT sensor data for early risk prediction of diabetes as an example of NCDs. In the

experiment, a verified diabetic dataset has been used for training, while the testing has been performed on an

artificially generated data collection from sensors

**TITLE 10 :**

Medical image analysis based on deep learning approach,2021, Multimedia Tools and Applications volume 80, pages24365–24398 (2021)

**AUTHORS:**

MuralikrishnaPuttagunta& S. Ravi[10]

**DESCRIPTION:**

Discussed are new algorithms and strategies in the area of deep learning. DLA can support clinical decisions for next-generation radiologists. DLA can automate radiologist workflow and facilitate decision-making for inexperienced radiologists. DLA is intended to aid physicians by automatically identifying and classifying lesions to provide a more precise diagnosis. DLA can help physicians to minimize medical errors and increase medical efficiency in the processing of medical image analysis.

# SYSTEM ANALYSIS

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

Disease prediction systems have been playing a significant role in the life of people, and it has been considered an important topic by many academics. Although prediction results achieved are promising, these traditional approaches are still far from being highly accurate and efficient. The existing systems are simple and effective but are extremely vulnerable to impact. Moreover, state-of-the-art methods can detect diseases pertaining to certain organs only while some severe conditions may go completely undetected. This could lead practitioners to false assumptions and improper diagnosis and treatments provided to patients

**3.2 PROPOSED SYSTEM**

We propose an end-to-end application that predicts Cancer, Diabetes, Heart disease using distinct techniques. The aim of developing this application with the help of Deep Learning algorithms is to immensely help to solve health-related issues by assisting the physicians to predict and diagnose diseases at an early stage. The accurate analysis by our proposed application benefits in early disease prediction, patient care, and community services. The proposed application is simple and it shows the good efficient performance. The overall accuracy of the proposed scheme has been evaluated with the traditional state-of-the-art models and the results from our proposed application show a higher accuracy rate. The proposed prediction model caters to this objective following a stepwise approach through cleaning, feature extraction, and classification.Initially disease dataset is taken as an input for the system for Diabetes, heart disease, and cancer datasets.Relevant elements are fed to the neural network by eliminating the irrelevant features using feature selectionAs the algorithms are selected, the symptoms are processed, and the most likely disease is given as an output. The performance of the network is dependent on the number of guidelines used which decide the behavior of the network.The input is given to the neurons which perform some operation to generate the output

**3.3 REQUIREMENT SPECIFICATION**

**3.3.1 HARDWARE REQUIREMENTS**

Processor : Pentium Dual Core 2.00GHZ

Hard disk : 120 GB

RAM : 2GB (minimum)

Keyboard : 110 keys enhanced

**3.3.2 SOFTWARE REQUIREMENTS**

Operating system : Windows7 (with service pack 1), 8, 8.1,10

IDE : Anaconda

Language : Python

**3.4 PLATFORM SPECIFICATION– ANACONDA**

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field.Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you’re using Anaconda.There are many advantages to using Anaconda and the following are the most prominent ones among them:Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so you don’t face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn’t be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues.It gives you a seamless environment that simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it’s a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

**How to Use Anaconda for Python**

Now that we have discussed all the basics in our Python Anaconda tutorial, let’s discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you’d need to see how many Conda environments are present in your machine.

condaenv list

It will list all the available Conda environments in your machine.

Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command:

conda create -n <your\_environment\_name>

You can replace <your\_environment\_name> with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

proceed ([y])/n)?

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

conda create -n <your\_environment\_name> python=3.6

Similarly, if you want to create an environment with a particular package, you can use the following command:

conda create -n <your\_environment\_name>pack\_name

Here, you can replace pack\_name with the name of the package you want to use.

If you have a .yml file, you can use the following command to create a new Conda environment based on that file:

condaenv create -n <your\_environment\_name> -f <file\_name>.yml

We have also discussed how you can export an existing Conda environment to a .yml file later in this article.

**Activating an Environment**

You can activate a Conda environment by using the following command:

conda activate <environment\_name>

You should activate the environment before you start working on the same. Also, replace the term <environment\_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command:

conda deactivate

**Installing Packages in an Environment**

Now that you have an activated environment, you can install packages into it by using the following command:

conda install <pack\_name>

Replace the term <pack\_name> with the name of the package you want to install in your Conda environment while using this command.

**Updating Packages in an Environment**

If you want to update the packages present in a particular Conda environment, you should use the following command:

conda update

The above command will update all the packages present in the environment. However, if you want to update a package to a certain version, you will need to use the following command:

conda install <package\_name>=<version>

**Exporting an Environment Configuration**

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yml file and share it with that person. Now, they can create an environment like your one by using the .yml file.

For exporting the environment to the .yml file, you’ll first have to activate the same and run the following command:

condaenv export ><file\_name>.yml

The person you want to share the environment with only has to use the exported file by using the ‘Creating a New Environment’ command we shared before.

**Removing a Package from an Environment**

If you want to uninstall a package from a specific Conda environment, use the following command:

conda remove -n <env\_name><package\_name>

On the other hand, if you want to uninstall a package from an activated environment, you’d have to use the following command:

conda remove <package\_name>

**Deleting an Environment**

Sometimes, you don’t need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

condaenv remove –name <env\_name>

The above command would delete the Conda environment right away.

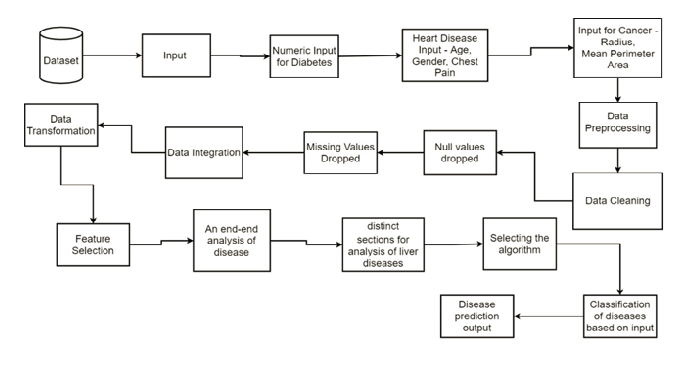
# SYSTEM DESIGN

**CHAPTER 4**

**SYSTEM DESIGN**

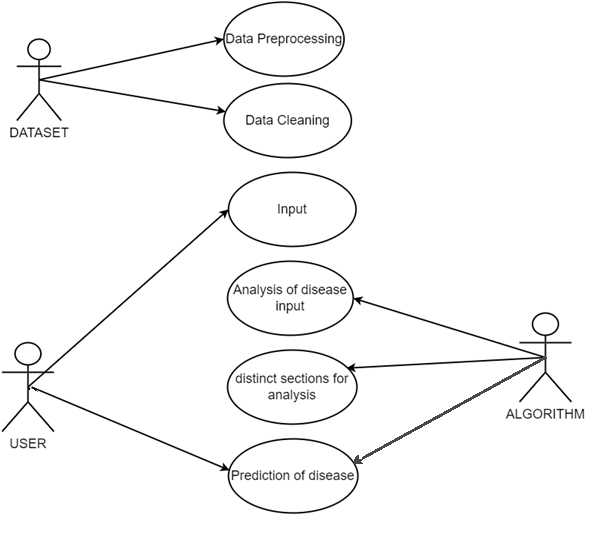
**4.1 SYSTEM ARCHITECTURE**

This diagram is nothing but a simple description of all the entities that have been incorporated into the system. The diagram represents the relations between each of them and involves a sequence of decision-making processes and steps. You can simply call it a visual or the whole process and its implementation. All functional correspondences are explained in this diagram.



**Fig 4.1- Architecture Diagram**

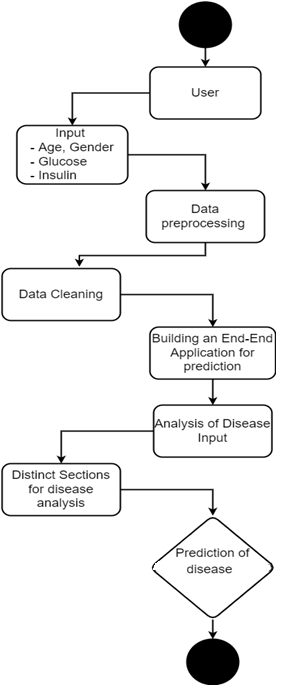
**4.2 USE-CASE DIAGRAM**

A use case diagram is a standard diagram that shows all interactions between the user, dataset, and algorithm used. It is developed in the early stages of the process.

**Fig 4.2- Use-Case Diagram**

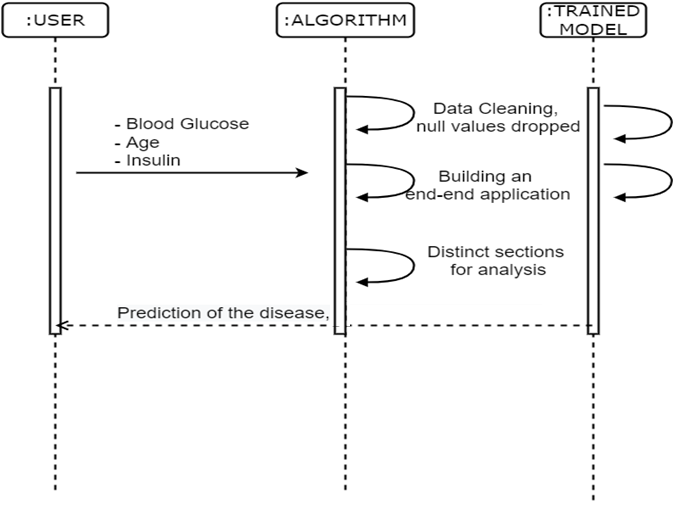
**4.3 ACTIVITY DIAGRAM**

In simple terms, a diagram that represents the order of all activities is called the activity diagram. It shows the workflow between different activities that take place in the whole process. However, these are not exactly flowcharts but are similar.



**Fig 4.3 - Activity Diagram**

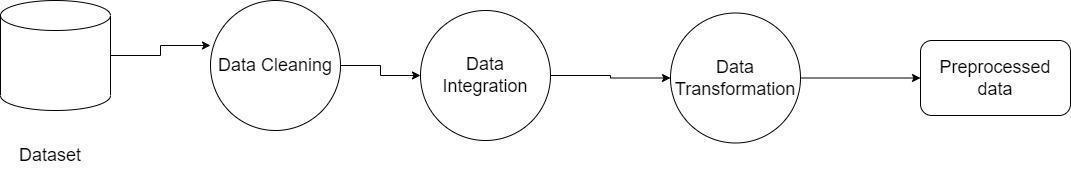
**4.4 SEQUENCE DIAGRAM**

These are other kinds of interaction-based diagrams that show how all the operations are carried out. They capture the context of collaborations between objects and processes.

**Fig 4.4 - Sequence Diagram**

**4.5 DATA FLOW DIAGRAM LEVEL 0**

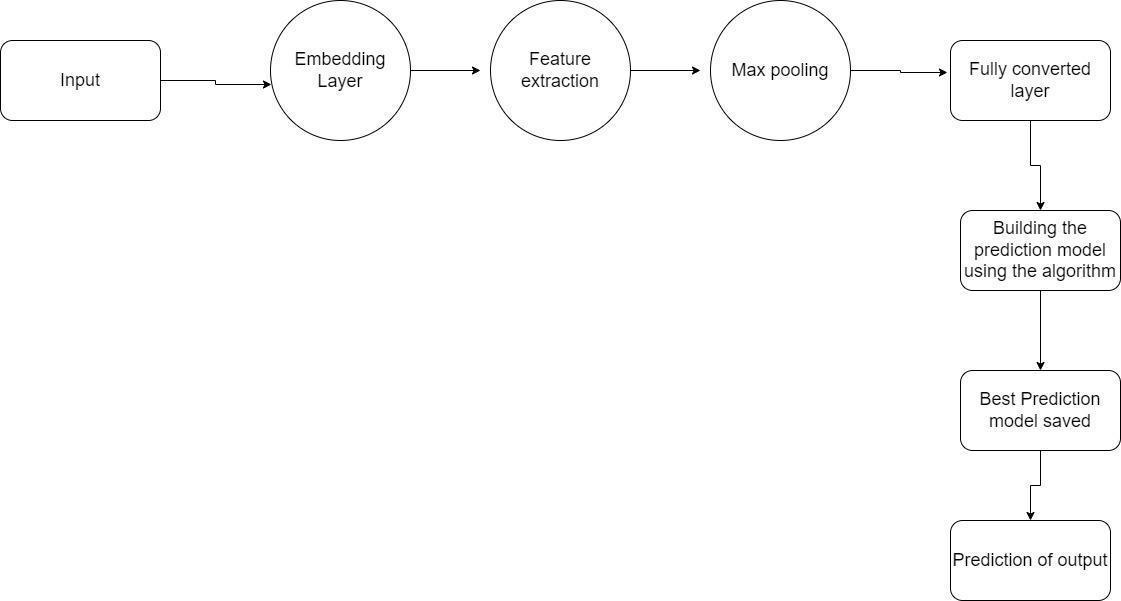
This is basically a contextual diagram, also referred to as a “context diagram”. It only represents the top level or the 0 Level in the whole process.it gives an abstraction kind of view and shows the whole system as a single process and its relationship to externalities.



**Fig 4.5 - Data Flow Diagram 0**

**4.6 DATA FLOW DIAGRAM LEVEL 1**

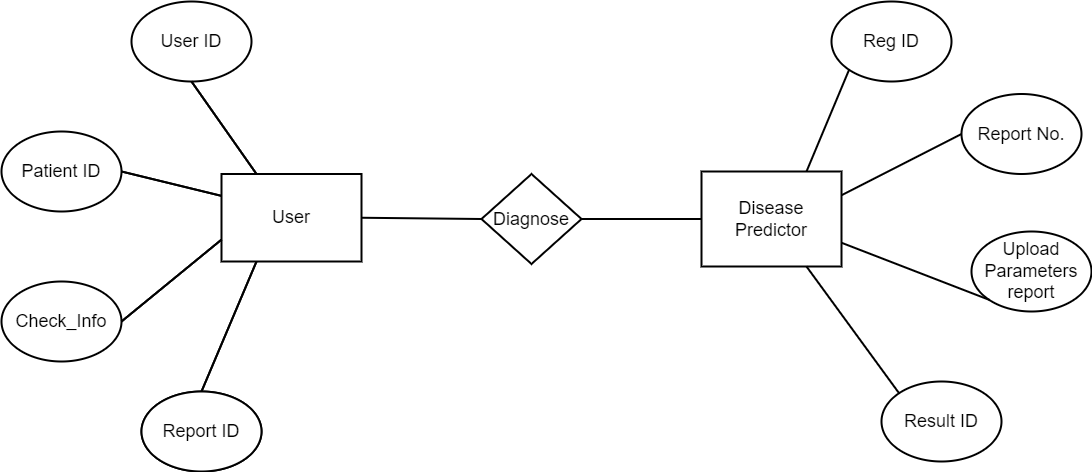
Level DFDs represent the complete system as a single process. itrotates every process and subprocess that comes together in a sequence to form the complete system. This along with ) and 2-level data flow diagrams comprise the “fundamental system model”.



**Fig 4.6 - Data Flow Diagram 1**

**4.7 ER DIAGRAM**

ER stands for Entity Relationship. These diagrams display the relationship of entities that are used and stored in the database. They explain the structure of the whole process. These diagrams can be made using three basic concepts, attributes, relationships, and entities.



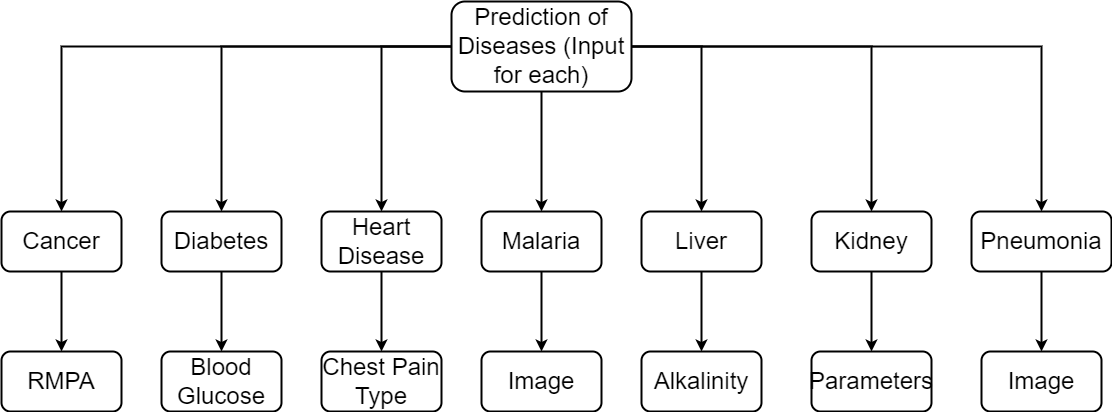
**Fig 4.7 - ER Diagram**

# MODULE DESCRIPTION

**CHAPTER 5**

**MODULE DESCRIPTION**

**5.1 Input Data:**

****

**Fig 5.1 Input values for all diseases**

Initially, the disease dataset is taken as an input for the system for Diabetes, heart disease, and cancer datasets. Relevant elements are fed to the neural network by eliminating the irrelevant features using feature selection. The input is given to the network which performs some operations to generate the output. Inputs for various diseases are given as below –

Prediction of cancer disease via perimeters such as radius mean perimeter area

Prediction of Diabetes via blood glucose, thickness (numeric values)

Prediction of Heart disease via - chest pain type, age, gender (numeric)

To predict Malaria and Pneumonia we get the Image as input.

Prediction of liver diseases - Input

Age

Gender

Alkalinity

Phosphate balance

Prediction of Kidney Diseases - Input

Input is obtained by fetching machine-oriented data from the dataset that is used to analyze the condition of the kidneys.

Prediction of Pneumonia – Input

Image is used as an input, image processing is leveraged.

Prediction of Malaria – Input

Image is used as an input, image processing is leveraged.

**5.2 Cancer, Diabetes Heart Disease Prediction**

The diagnosis of heart disease depends on the detailed and precise analysis of the patient's clinical test data and an individual's health history.

Here, we’re using the Deep Neural Network, DNN algorithm to diagnose heart diseases, diabetes, and cancer-related ailments.

The DNN with multiple hidden layers is used hence the proposed model has high performance than ANN.

Input for Diabetes(Numeric Values)

Glucose: the concentration test in glucose

Skin thickness: the thickness of the skin in the triceps fold (mm)

Insulin: insulin serum for 2-h (mu U/ml)

Input for Heart Diseases

Age: current age of the patient

Gender

Chest Pain Type

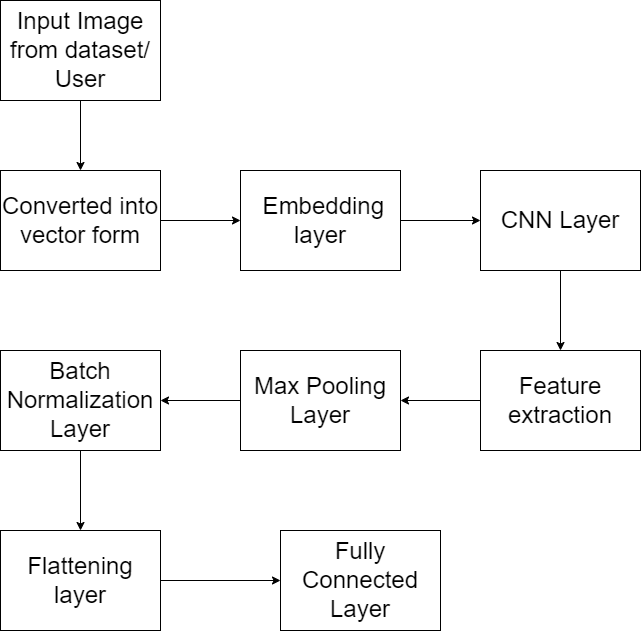
Input For Cancer-related Ailments

Radius

Mean Perimeter Area

**5.3 CNN for Image Classification**

In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyze visual imagery. It is a type of artificial neural network used primarily for image recognition and processing, due to its ability to recognize patterns in images. While CNNs are designed to solve problems with visual imagery, they also have many applications outside of image recognition and analysis, including image classification, natural language processing, drug discovery, and health risk assessments. CNNs also help provide depth estimation for self-driving cars. A CNN is a powerful tool but requires millions of labeled data points for training. Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and output an activation value. The features are converted to a new representation called word embedding by the layer called Embedding Layer. The Embedding layer's output is given to the CNN layer for feature extraction. In this model, max-pooling has been used. In addition, the batch normalization layer has been used as it involved the training of a very deep neural network. So the technique adjusts the scaling and activation to normalize the input layer and speed up the learning procedure between hidden units. Towards the last stage of the CNN used in the study, there is a flatting layer to convert the output of convolutional layers into a single-dimensional feature vector. In a fully connected layer, each neuron of the previous layer is directly connected to each of the neurons in its next layer. The fully connected network with dense layers finally drives the classification decision.

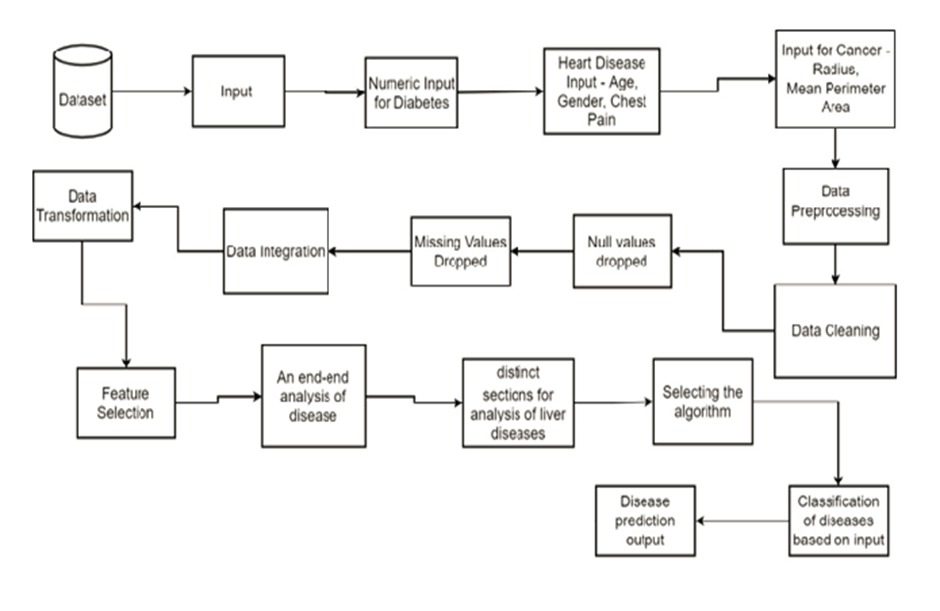


**Fig 5.2 CNN Architecture**

**5.4 Logistic Regression for Numeric Input**

Logistic regression is an algorithm used to predict the probability of categorical dependent variables. The dataset imported will be divided into two parts, namely training data and testing data. Training data is used as a basis for building models. This type of analysis is used to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation. In this way, you can predict how likely it is that an event will happen or a choice will be made. In order to predict the output of a categorical dependent variable, logistic regression must first determine the category or class of the dependent variable. This can be accomplished through a simple calculation, which returns an estimate of how likely it is that any given value within that category(s) will occur. The probabilistic values returned by this equation are then used to generate a prediction for how different combinations of independent variables would produce this particular output. Meanwhile, testing data is used as a basis for testing or validating the model. The data should contain different variables like age, gender, sex, chest pain, glucose levels, insulin levels, etc. In logistic regression, the dependent variable is always binary. Prediction using the logistic regression method will produce several data that can be used as a basis for concluding to make predictions. We have used a sigmoid function which helps in the graphical representation of the classified data. By using logistic regression, the accuracy is increased.

**5.5 End to End Diagnosis**



**Fig 5.3 Flow of End - End diagnosis**

For the analysis of all these diseases, we will create an end-to-end application that is able to analyze all of the input data for the diseases, process the data using distinct techniques, and then accurately predict the most likely disease which is given as output. The application will contain distinct sections for analysis of liver diseases, cardiovascular diseases, kidney-related ailments, cancer, pneumonia, and malaria. Not only just the disease, but the application would also predict the most likely disease based on the given input data. The application will be designed to predict the disease, mention the aftermath of the same, how can it be treated and also the factors or changes that caused the disease. The application will act as a virtual assistant in major clinical laboratories, healthcare centers and medical clinics. The proposed model is a dynamic web application for an end to end disease prediction and diagnosis.

The application has been designed using the Flask interface that would effectively display the output and brief out the symptoms. The complete check-up would be performed using all the parameters and images as a data set. In future, with a little improvement, It would fetch data from the cloud and can be retrieved whenever needed.Our proposed methodology helps to improve the accuracy of diagnosis and is greatly helpful for further treatment. This work will be useful for identifying the patients who suffer from heart disease. When a patient is predicted with a positive result their reports and data can be closely analyzed.

# IMPLEMENTATION

**CHAPTER 6**

**IMPLEMENTATION**

**GUI & Preprocessing Module**

This is the interface where the user or patient interacts with the application. This UI is built in a simple way so that it will be easy for the users to interact with.. In this module, these input parameters are trained well and a model is created. These input parameters have to be trained well in order to obtain a high accuracy The features in the training data and the quality of labeled training data will determine how accurately the machine learns to identify the outcome. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset, and so the data preprocessing module is required to perform tasks such as cleaning the data and making it suitable for a machine learning model which helps in increasing the accuracy and efficiency of a machine learning model. To accurately predict the case using ML we need our data to be cleaned and in a formatted way. In data processing step we also divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model Data preprocessing comprises of 4 steps.

1. The first step is the data cleaning in which the duplicate, incorrectly formatted, corrupted data will be fixed or removed.
2. The second step is the data integration is which it combines multiple sources data into single view.
3. The third step is the data reduction step in which the data are encoded, scaled sorted if needed.
4. The final step is the data transformation is which the data is transformed into a required format

**Implementation of CNN**

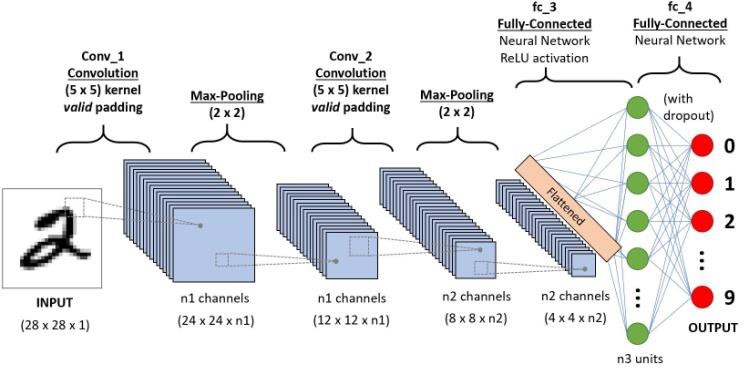
CNN Classification involves 4 steps namely Convolution layer, Activation function, max Pooling, Fully connected network.

**Relu layer**: Rectified linear unit is used to scale the parameters to non negative values. We get pixel values as negative values too. In this layer we make them as 0’s. The purpose of applying the rectifier function is to increase the non-linearity in our images (fig 4.1.4). The reason we want to do that is that images are naturally non-linear. The rectifier serves to break up the linearity even further in order to make up for the linearity that we might impose an image when we put it through the convolution operation. What the rectifier function does to an image like this is remove all the black elements from it, keeping only those carrying a positive value (the grey and white colors).The essential difference between the non-rectified version of the image and the rectified one is the progression of colors. After we rectify the image, you will find the colors changing more abruptly. The gradual change is no longer there. That indicates that the linearity has been disposed of.

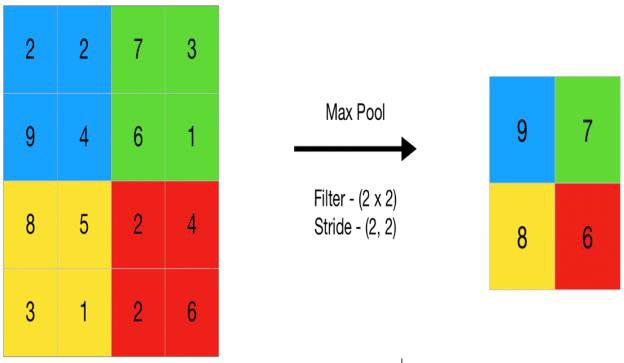
**Pooling layer**: The pooling (POOL) layer reduces the height and width of the input. It helps reduce computation, as well as helps make feature detectors more invariant to its position in the input.This process is what provides the convolutional neural network with the “spatial variance” capability (fig 4.1.5). In addition to that, pooling serves to minimize the size of the images as well as the number of parameters which, in turn, prevents an issue of “overfitting” from coming up. Overfitting in a nutshell is when you create an excessively complex model in order to account for the idiosyncracies we just mentioned. The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model’s invariance to local translation.

**Fully Connected layer**: The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network. It has neurons linked to each other, and activates if it identifies patterns and sends signals to output layer .the output layer gives output class based on weight values.

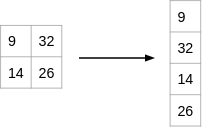
In Convolution layer a 3\*3 kernel is applied on the input matrix with stride 1, so that the size of the image is reduced. The obtained matrix is sent to the activation function layer, here ReLU(Rectified Linear Unit)activation function is used to eliminate negative values (fig 4.1.4). The next step is max pooling, here the rectified map goes through a pooling layer, pooling is down sampling operation that reduces the dimensionality of feature map. The output of this layer is a matrix with reduced size containing all the features. These 3 steps are repeated for two times, in which 32 filters are used each time. The next step is Flattering in which we convert the 2D array from pooling into a long continuous linear vector to which a filter of size 64 is applied followed by activation function. Later a Drop Out of size 0.5 is applied to prevent over-fitting. In the next step ANN model is applied, in which sigmoid activation is used function to detect the presence of stone.



**Fig: 6.1 Convolution methodology**

****

**Fig: 6.2Maxpooling**



**Fig: 6.3Converting matrix into linear array form**

# 

# TESTING

**CHAPTER 7**

**TESTING**

**Testing Methodologies**

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

**7.1 Unit Testing**

A unit is the smallest possible testable software component. A unit can be function or procedure implemented in a procedural programming languages. A unit may also be a small-sized COTS component purchased from an outside vendor that is undergoing evaluation by the purchaser, or a simple module retrieved from an in-house reuse library. Unit test results are recorded for future testing process. This result document used for integration and system tests.

Some of the phases for unit test planning are;

* Describe Unit Test Approach and Risks.
* Identify Unit features to be tested.
* Add levels of detail to the test plan.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST NO.** | **INPUT** | **EXPECTED BEHAVIOUR** | **TESTING** | **STATUS** |
| 1 | User uploading proper input values | Data updated to the trained model | Unit Testing | Pass |
| 2 | User uploading invalid input data | Data not updated to the trained model | Unit Testing | Fail |
| 3 | User uploading proper image | Disease predicted | Output Testing | Pass |
| 4 | User uploading invalid image | Disease prediction model not loaded | Output Testing | Fail |

# CONCLUSION

**CONCLUSION AND FUTURE ENHANCEMENTS**

The results thus justify the effectiveness of the proposed method in disease prediction. Although the results are promising, the datasets being used are relatively smaller in comparison to the big data culture predominant in the present day and age. Thus, the research’s future path will be to test the proposed model on large-high-quality datasets to check the effectiveness and reliability of the proposed model. Our proposed methodology helps to improve the accuracy of diagnosis and is greatly helpful for further treatment. This work will be useful for identifying the patients who suffer from heart disease. When a patient is predicted with a positive result their reports and data can be closely analyzed. In future enhancements, the accuracy has to be tested with different datasets and to apply other AI algorithms to check the accuracy estimation.

# APPENDICES

**APPENDIX 1**

#from flask\_wtf import FlaskForm

#from wtforms import StringField, SubmitField

"""

classDiabetesCheck(FlaskForm):

pregnancies = StringField("Pregnancies")

glucose = StringField("Glucose")

pregnancies = StringField("BloodPressure")

glucose = StringField("SkinThickness")

pregnancies = StringField("Insulin")

bmi = StringField("BMI")

diabetesPedigreeFunction = StringField("DiabetesPedigreeFunction")

age = StringField("Age")

submit= SubmitField("Check It !")

classCancerCheck(FlaskForm):

radius\_mean = StringField("'Radius\_mean'")

texture\_mean = StringField("Texture\_mean")

perimeter\_mean = StringField("Perimeter\_mean")

area\_mean = StringField("Area\_mean")

smoothness\_mean = StringField("Smoothness\_mean")

compactness\_mean = StringField("Compactness\_mean")

concavity\_mean = StringField("Concavity\_mean")

concave\_points\_mean = StringField("concave points\_mean")

symmetry\_mean = StringField("symmetry\_mean")

fractal\_dimension\_mean = StringField("fractal\_dimension\_mean")

radius\_se = StringField("radius\_se")

texture\_se = StringField("texture\_se")

perimeter\_se = StringField("perimeter\_se")

area\_se = StringField("area\_se")

smoothness\_se = StringField("smoothness\_se")

compactness\_se = StringField("compactness\_se")

concavity\_se = StringField("concavity\_se")

concave\_points\_se = StringField("concave points\_se")

smoothness\_se = StringField("symmetry\_se")

fractal\_dimension\_se = StringField("fractal\_dimension\_se")

radius\_worst = StringField("radius\_worst")

texture\_worst = StringField("texture\_worst")

parameter\_worst = StringField("perimeter\_worst")

area\_worst = StringField("area\_worst")

smoothness\_worst = StringField("smoothness\_worst")

compactness\_worst = StringField("compactness\_worst")

concavity\_worst = StringField("concavity\_worst")

concave\_points\_worst = StringField("concave points\_worst")

symmetry\_worst = StringField("symmetry\_worst")

fractal\_dimension\_worst = StringField("fractal\_dimension\_worst")

submit= SubmitField("Check It !")

#Important Modules

from flask import Flask,render\_template, url\_for ,flash , redirect

fromsklearn.externals import joblib

from flask import request

importnumpy as np

importtensorflow

#from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Input, Flatten, SeparableConv2D

#from flask\_sqlalchemy import SQLAlchemy

#from model\_class import DiabetesCheck, CancerCheck

#from tensorflow.keras.models import Sequential

#from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Input, Flatten, SeparableConv2D

#from tensorflow.keras.layers import GlobalMaxPooling2D, Activation

#from tensorflow.keras.layers.normalization import BatchNormalization

#from tensorflow.keras.layers.merge import Concatenate

#from tensorflow.keras.models import Model

importos

from flask import send\_from\_directory

fromtensorflow.keras.models import load\_model

fromtensorflow.keras.preprocessing import image

importtensorflow as tf

#from this import SQLAlchemy

app=Flask(\_name\_,template\_folder='template')

# RELATED TO THE SQL DATABASE

app.config['SECRET\_KEY'] = '5791628bb0b13ce0c676dfde280ba245'

#app.config["SQLALCHEMY\_DATABASE\_URI"] = "sqlite:///site.db"

#db=SQLAlchemy(app)

#from model import User,Post

#//////////////////////////////////////////////////////////

dir\_path = os.path.dirname(os.path.realpath(\_file\_))

# UPLOAD\_FOLDER = dir\_path + '/uploads'

# STATIC\_FOLDER = dir\_path + '/static'

UPLOAD\_FOLDER = 'uploads'

STATIC\_FOLDER = 'static'

#graph = tf.get\_default\_graph()

#with graph.as\_default():;

fromtensorflow.keras.models import load\_model

model = load\_model('model 111.h5')

model222=load\_model("my\_model.h5")

#FOR THE FIRST MODEL

# call model to predict an image

defapi(full\_path):

data = image.load\_img(full\_path, target\_size=(50, 50, 3))

data = np.expand\_dims(data, axis=0)

data = data \* 1.0 / 255

#with graph.as\_default():

predicted = model.predict(data)

return predicted

#FOR THE SECOND MODEL

def api1(full\_path):

data = image.load\_img(full\_path, target\_size=(64, 64, 3))

data = np.expand\_dims(data, axis=0)

data = data \* 1.0 / 255

#with graph.as\_default():

predicted = model 222.predict(data)

return predicted

# home page

#@app.route('/')

#def home():

# returnrender\_template('index.html')

# processing uploaded file and predict it

@app.route('/upload', methods=['POST','GET'])

defupload\_file():

ifrequest.method == 'GET':

returnrender\_template('index.html')

else:

try:

file = request.files['image']

full\_name = os.path.join(UPLOAD\_FOLDER, file.filename)

file.save(full\_name)

indices = {0: 'PARASITIC', 1: 'Uninfected', 2: 'Invasive carcinoma', 3: 'Normal'}

result = api(full\_name)

print(result)

predicted\_class = np.asscalar(np.argmax(result, axis=1))

accuracy = round(result[0][predicted\_class] \* 100, 2)

label = indices[predicted\_class]

returnrender\_template('predict.html', image\_file\_name = file.filename, label = label, accuracy = accuracy)

except:

flash("Please select the image first !!", "danger")

return redirect(url\_for("Malaria"))

@app.route('/upload11', methods=['POST','GET'])

def upload11\_file():

ifrequest.method == 'GET':

returnrender\_template('index2.html')

else:

try:

file = request.files['image']

full\_name = os.path.join(UPLOAD\_FOLDER, file.filename)

file.save(full\_name)

indices = {0: 'Normal', 1: 'Pneumonia'}

result = api1(full\_name)

if(result>50):

label= indices[1]

accuracy= result

else:

label= indices[0]

accuracy= 100-result

returnrender\_template('predict1.html', image\_file\_name = file.filename, label = label, accuracy = accuracy)

except:

flash("Please select the image first !!", "danger")

return redirect(url\_for("Pneumonia"))

@app.route('/uploads/<filename>')

defsend\_file(filename):

returnsend\_from\_directory(UPLOAD\_FOLDER, filename)

#//////////////////////////////////////////////

#app.config["SQLALCHEMY\_DATABASE\_URI"] = "sqlite:///site.db"

#db=SQLAlchemy(app)

#class User(db.Model):

## username = db.Column(db.String(20), unique=True, nullable=False)

# email = db.Column(db.String(120), unique=True, nullable=False)

#image\_file = db.Column(db.String(20), nullable=False, default='default.jpg')

# password = db.Column(db.String(60), nullable=False)

#posts = db.relationship('Post', backref='author', lazy=True)

#def \_repr\_(self):

# return f"User('{self.username}', '{self.email}', '{self.image\_file}')"

@app.route("/")

@app.route("/home")

def home():

returnrender\_template("home.html")

@app.route("/about")

def about():

returnrender\_template("about.html")

@app.route("/cancer")

def cancer():

returnrender\_template("cancer.html")

@app.route("/diabetes")

def diabetes():

#if form.validate\_on\_submit():

returnrender\_template("diabetes.html")

@app.route("/heart")

def heart():

returnrender\_template("heart.html")

@app.route("/liver")

def liver():

#if form.validate\_on\_submit():

returnrender\_template("liver.html")

@app.route("/kidney")

def kidney():

#if form.validate\_on\_submit():

returnrender\_template("kidney.html")

@app.route("/Malaria")

def Malaria():

returnrender\_template("index.html")

@app.route("/Pneumonia")

def Pneumonia():

returnrender\_template("index2.html")

"""

@app.route("/register", methods=["GET", "POST"])

def register():

form =RegistrationForm()

ifform.validate\_on\_submit():

#flash("Account created for {form.username.data}!".format("success"))

flash("Account created","success")

return redirect(url\_for("home"))

returnrender\_template("register.html", title ="Register",form=form )

@app.route("/login", methods=["POST","GET"])

def login():

form =LoginForm()

ifform.validate\_on\_submit():

#if form.email.data =="sho" and form.password.data=="password":

flash("You Have Logged in !","success")

return redirect(url\_for("home"))

#else:

# flash("Login Unsuccessful. Please check username and password``,''danger``)

returnrender\_template("login.html", title ="Login",form=form )

def ValuePredictor1(to\_predict\_list):

to\_predict = np.array(to\_predict\_list).reshape(1,30)

loaded\_model = joblib.load("model")

result = loaded\_model.predict(to\_predict)

return result[0]

@app.route('/result1',methods = ["GET","POST"])

def result():

ifrequest.method == 'POST':

to\_predict\_list = request.form.to\_dict()

to\_predict\_list=list(to\_predict\_list.values())

to\_predict\_list = list(map(float, to\_predict\_list))

result = ValuePredictor(to\_predict\_list)

if int(result)==1:

prediction='cancer'

else:

prediction='Healthy'

return(render\_template("result.html", prediction=prediction))"""

defValuePredictor(to\_predict\_list, size):

to\_predict = np.array(to\_predict\_list).reshape(1,size)

if(size==8):#Diabetes

loaded\_model = joblib.load("model1")

result = loaded\_model.predict(to\_predict)

elif(size==30):#Cancer

loaded\_model = joblib.load("model")

result = loaded\_model.predict(to\_predict)

elif(size==12):#Kidney

loaded\_model = joblib.load("model3")

result = loaded\_model.predict(to\_predict)

elif(size==10):

loaded\_model = joblib.load("model4")

result = loaded\_model.predict(to\_predict)

elif(size==11):#Heart

loaded\_model = joblib.load("model2")

result =loaded\_model.predict(to\_predict)

return result[0]

@app.route('/result',methods = ["POST"])

def result():

ifrequest.method == 'POST':

to\_predict\_list = request.form.to\_dict()

to\_predict\_list=list(to\_predict\_list.values())

to\_predict\_list = list(map(float, to\_predict\_list))

if(len(to\_predict\_list)==30):#Cancer

result = ValuePredictor(to\_predict\_list,30)

elif(len(to\_predict\_list)==8):#Diabetes

result = ValuePredictor(to\_predict\_list,8)

elif(len(to\_predict\_list)==12):

result = ValuePredictor(to\_predict\_list,12)

elif(len(to\_predict\_list)==11):

result = ValuePredictor(to\_predict\_list,11)

#if int(result)==1:

# prediction ='diabetes'

#else:

# prediction='Healthy'

elif(len(to\_predict\_list)==10):

result = ValuePredictor(to\_predict\_list,10)

if(int(result)==1):

prediction='Sorry ! You are suffering Suffering'

else:

prediction='Congrats ! You are Healthy'

return(render\_template("result.html", prediction=prediction))

if \_name\_ == "\_main\_":

app.run(debug=True)

import pandas as pd

importnumpy as np

fromsklearn.preprocessing import StandardScaler

import random

fromsklearn.model\_selection import ShuffleSplit

fromsklearn.model\_selection import cross\_validate

fromimblearn.over\_sampling import SMOTE

fromsklearn.model\_selection import train\_test\_split

fromsklearn.linear\_model import LogisticRegression

fromsklearn.externals import joblib

data=pd.read\_csv("indian\_liver\_patient.csv")

data=data.fillna(method="ffill")

data.Gender=data.Gender.map({"Female":1,"Male":0})

data["Dataset"]=data["Dataset"].map({1:0,2:1})

np.random.shuffle(data.values)

print(data.columns)

target=data["Dataset"]

source=data.drop(["Dataset"],axis=1)

sm=SMOTE()

sc=StandardScaler()

lr=LogisticRegression()

source=sc.fit\_transform(source)

X\_train,X\_test,y\_train,y\_test= train\_test\_split(source,target,test\_size=0.01)

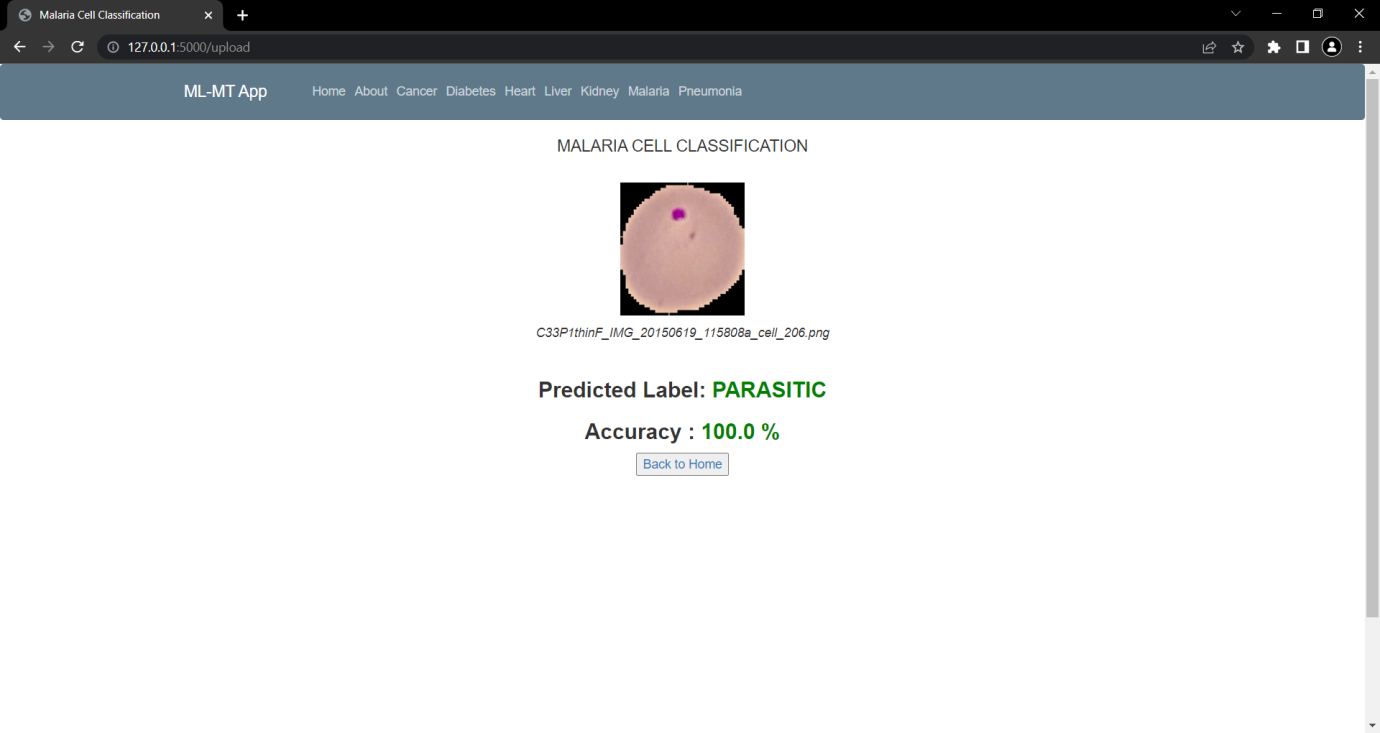
X\_train, y\_train=sm.fit\_sample(X\_train,y\_train)

cv=cross\_validate(lr,X\_train,y\_train,cv=10)

print(cv)

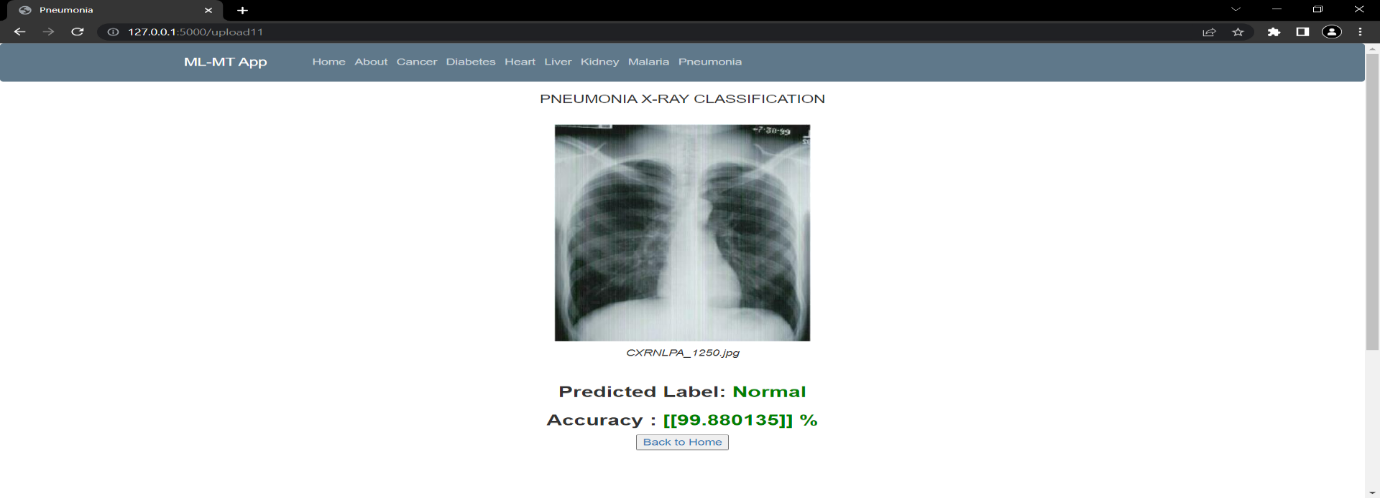
joblib.dump(lr,"model4"

**APPENDIX 2**

****

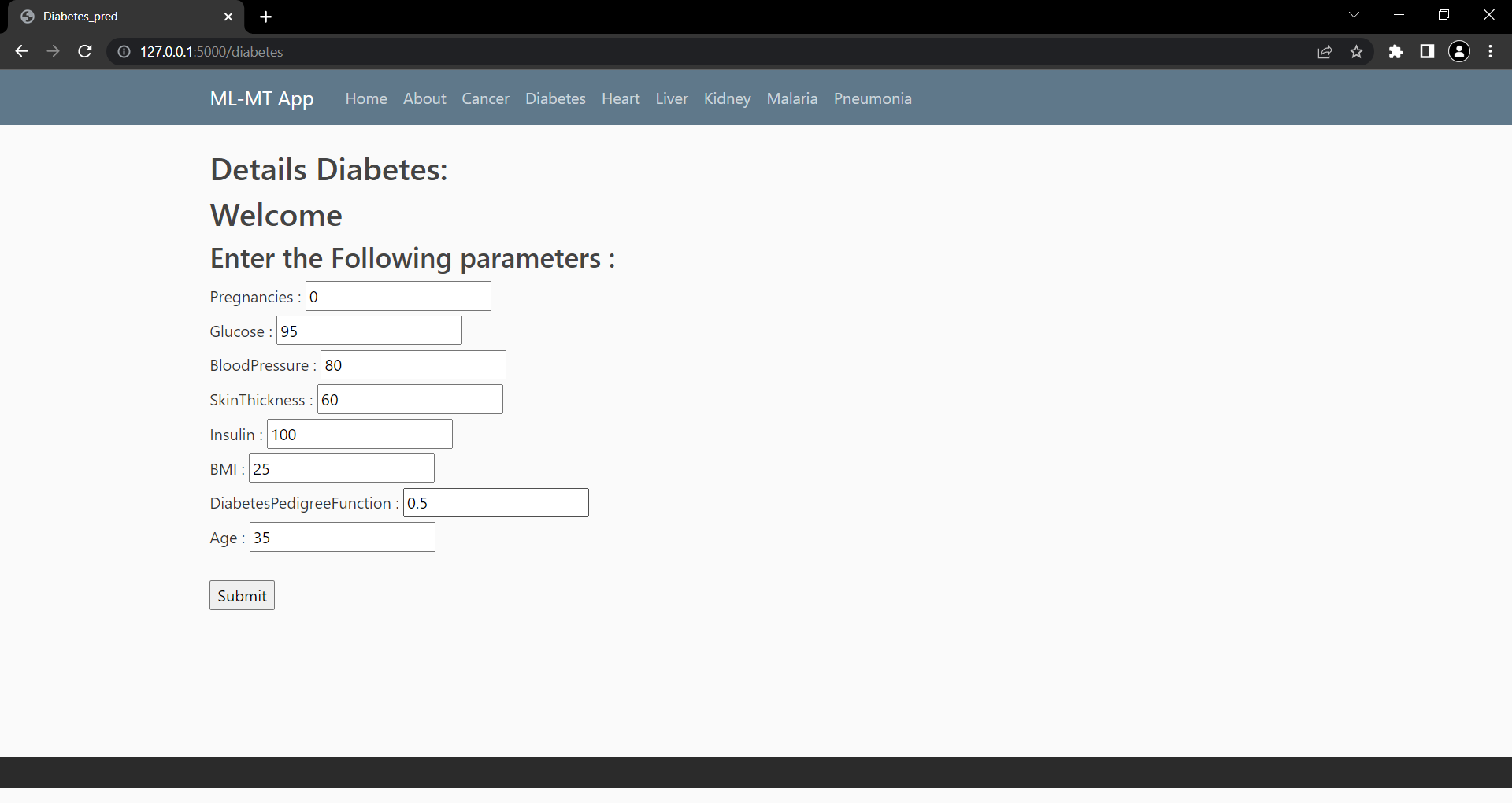
**Fig 9.1 Parasitic malaria predicted**

This is the image upload UI for malaria disease prediction system. The given image is predicted as Parasitic and the accuracy is found to be 100%

****

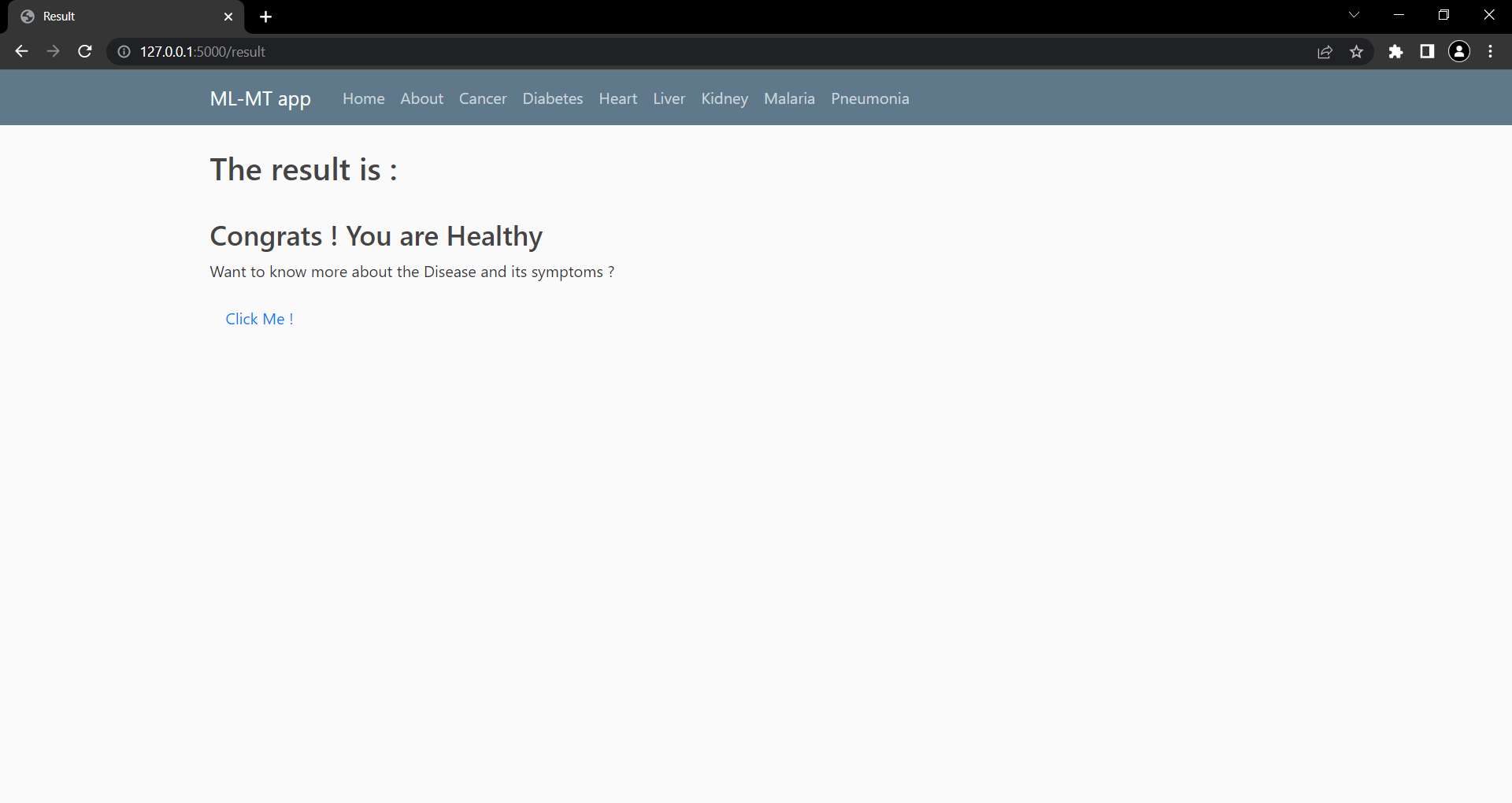
**Fig 9.2 Normal lungs Predicted**

This is the image upload UI for Lung disease system. The given image is predicted as normal so the lungs is not affected with any disease and the accuracy is found to be 99%

****

**Fig 9.3 Diabetes – input**

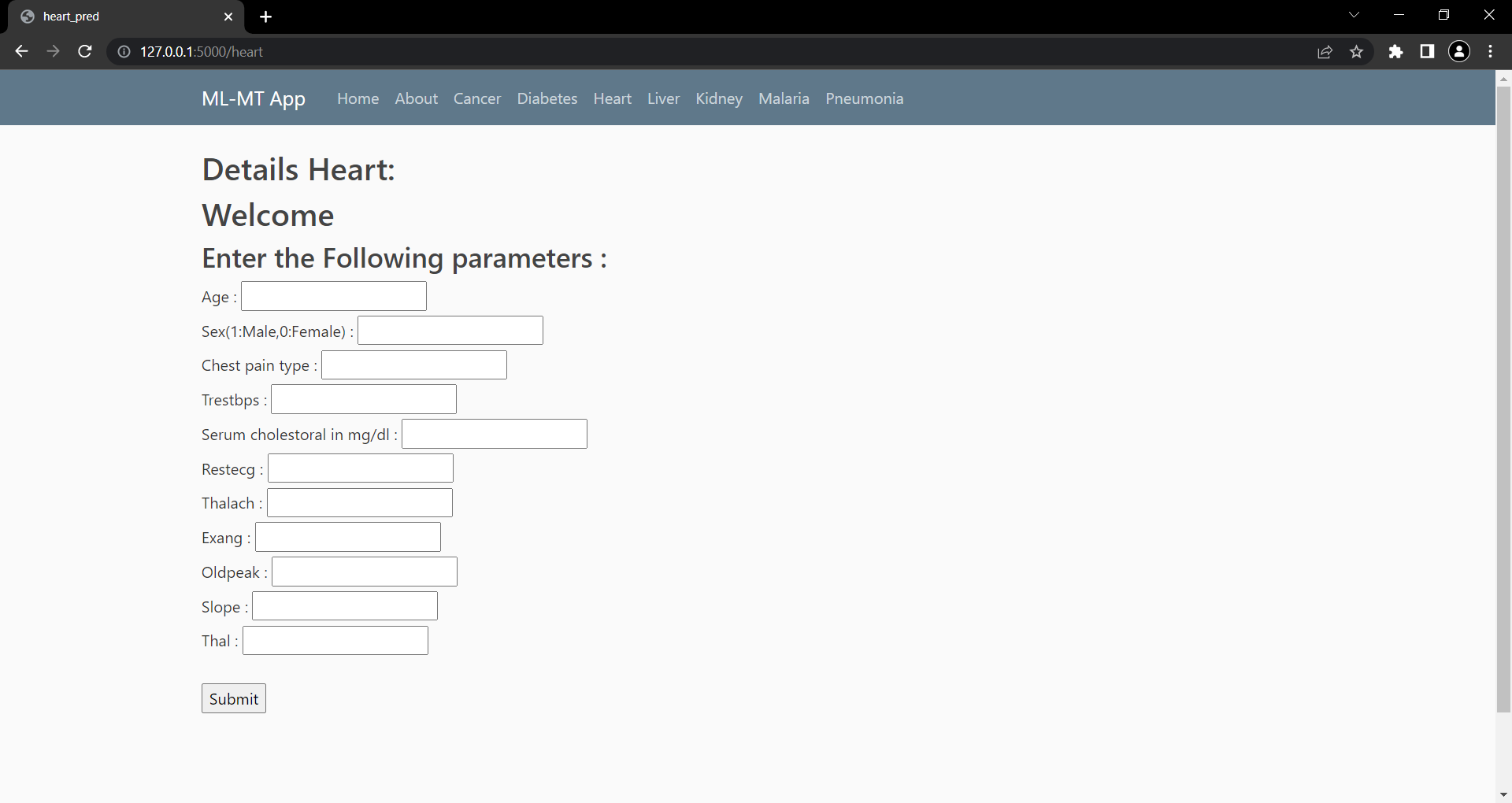
This is the input UI for diabetes prediction system. Each field has certain value and these values are passed to the trained ML model.

****

**Fig 9.4 Diabetes not predicted**

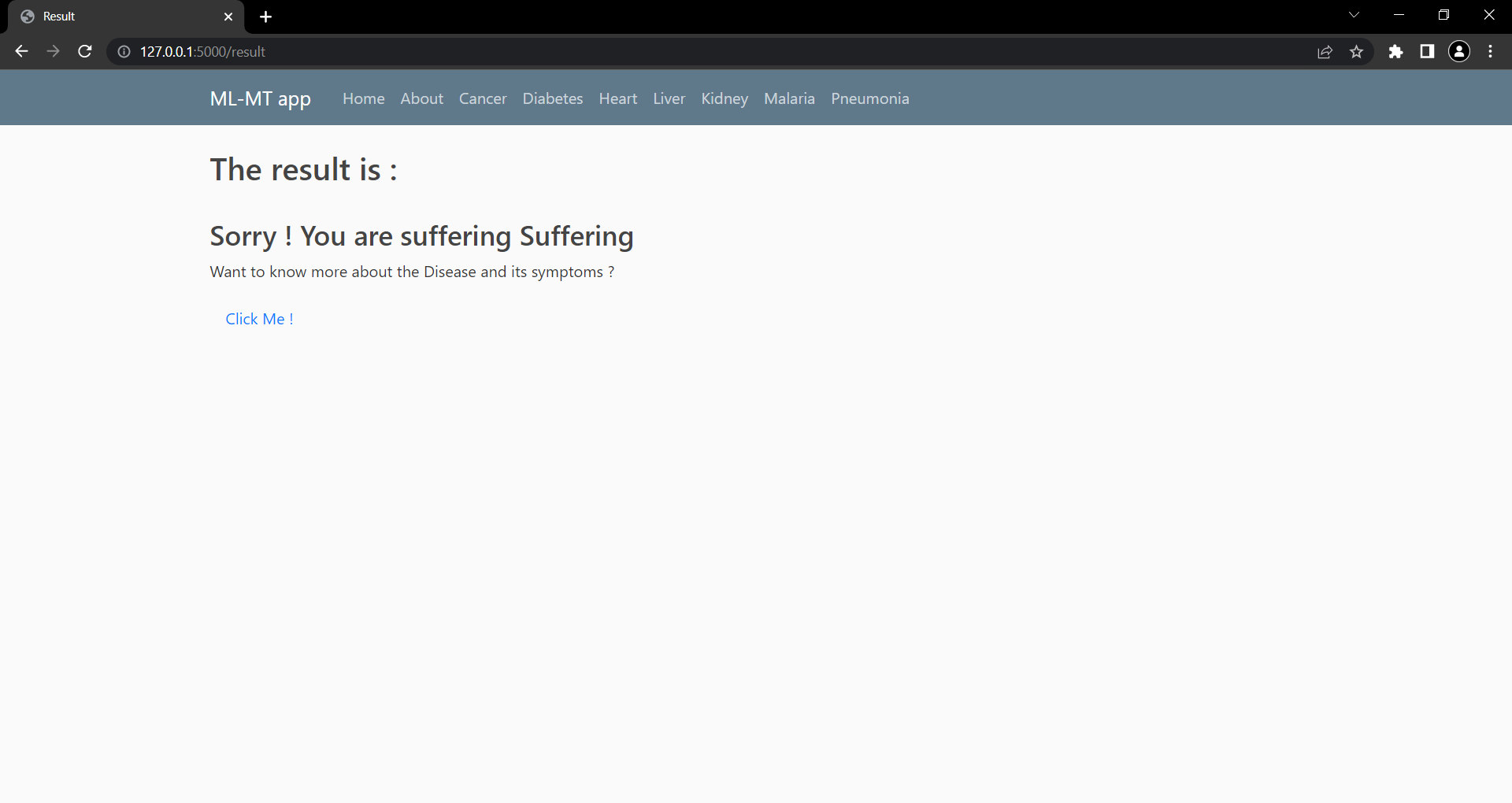
This is result page for Diabetes disease system. Upon submitting the data, the

Trained ML model predicts the output. Here there is no disease predicted for the given input values



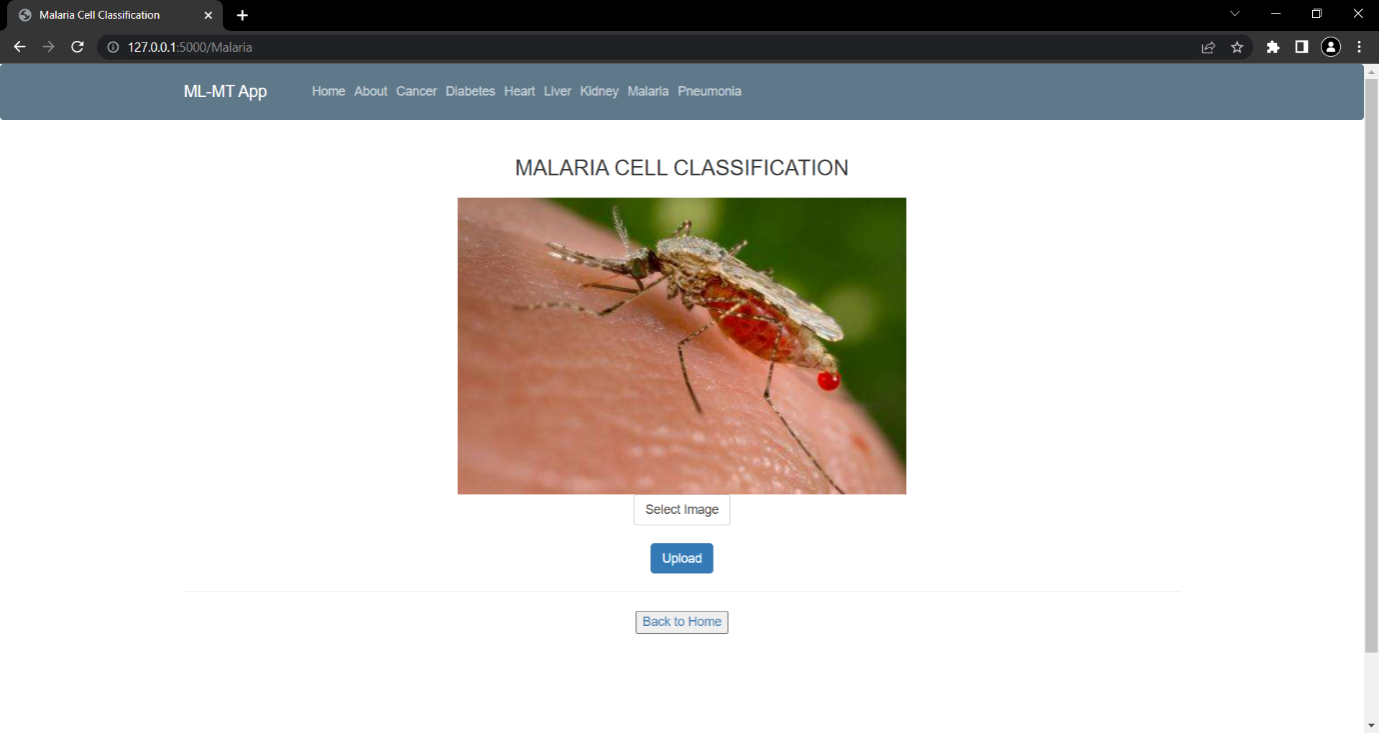
**Fig 9.5 Heart Disease Input**

This is the input UI for Heart Disease prediction system. Each field has certain value and these values are passed to the trained ML model.



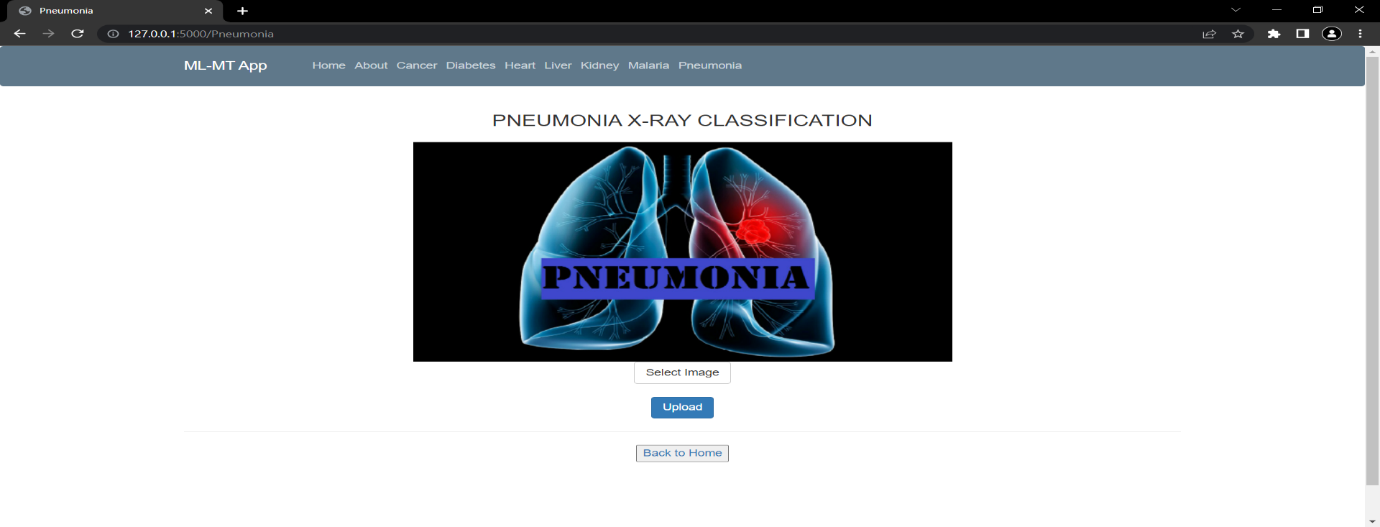
**Fig 9.6 Heart Disease Predicted**

This is result page for Heart disease prediction system. Upon submitting the data, the Trained ML model predicts the output. Here the heart disease is predicted for the given input values

****

**Fig 9.7 Malaria image input**

This is the image upload UI for Malaria disease prediction system. First, the input image has to be selected and then the upload button has to be clicked to upload the image.

****

**Fig 9.8 Pneumonia image input**

This is the image upload UI for Pneumonia disease prediction system. First, the input image has to be selected and then the upload button has to be clicked to upload the image.

**REFERENCE**

**REFERENCES**

[1] LivaFaes; Xiaoxuan Liu; Pearse A Kean, “A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis”, The Lancet Digital Health, Volume 1, Issue 6, October 2019

[2] M. Chen, Y. Hao, K. Hwang, L. Wang and L. Wang, "Disease prediction by machine learning over big data from healthcare communities", IEEE Access, vol. 5, no. 1, pp. 8869-8879, 2017

[3] Pushpa Singh; Narendra Singh; Krishna Kant Singh; Akansha Singh,” Diagnosing of disease using machine learning”,

2021, Machine Learning and the Internet of Medical Things in Healthcare Pages 89-111, 2021

[4] Alexander Selvikvag; Lundervold; Arvid  Lundervold; KatarzynaWęgrzyn-Wolska, “An overview of deep learning in medical imaging focusing on MRI”, ZeitschriftfürMedizinischePhysik, Volume 29, Issue 2, Pages 102-127, May 2019,

[5] Yuanyuan Pan; Minghuan Fu; Biao Cheng;  Xuefei Tao; Jing Guo,“Enhanced Deep Learning Assisted Convolutional Neural Network for Heart Disease Prediction on the Internet of Medical Things Platform, IEEE Access ( Volume: 8), May 2020,

[6] hirajDahiwade;  GajananPatle; EktaaMeshram,”Designing Disease Prediction Model Using Machine Learning Approach”,

2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 2019

[7] Durr-e-maknoonNisar; MeshrifAlruily ; Sultan H. Al Motor; Rashid Amin,“Healthcare Techniques Through Deep Learning:

Issues, Challenges and Opportunities”, 2019

[8] JeromeThevenot; Miguel Bordallo Lopez; AbdenourHadid,“A Survey On Computer Vision For Assistive

Medical Diagnosis From Faces”, IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 22, No. 5, 2018

[9) RahataraFerdousi; M. Anwar Hossain; Abdulmotaleb El Saddik, “Early-Stage Risk Prediction of Non-Communicable Disease Using Machine Learning in Health CPS”, IEEE Special Section On AI And IoT Convergence For Smart Health, July 2021

[10] Alice Othmania; Abdul RahmanTaleb; HazemAbdelkawy; AbdenourHadid, “Age estimation from faces using deep learning: a comparative analysis”, Computer Vision and Image Understanding, Volume 196,  102961, July 2020.