A Project Report on

**Lung Cancer Detection Using Image Segmentation**

*is submitted in partial fulfillment of the requirement for the award of the Degree of*

***BACHELOR OF TECHNOLOGY***

*to*

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR,**

**ANANTHAPURAMU**

**by**

|  |  |
| --- | --- |
| **P PRANOY DAVIS** | **(20AT1A05A6)** |
| **P PAVAN KUMAR REDDY** | **(21AT5A050A2)** |
| **L MANJUNATH NAIDU** | **(20AT1A0583)** |

**PD MAHAMMAD HUSSAIN (20AT1A0575)**

**Under the Guidance of**

**MR VIJAYA CHANDRA RAO M.Tech.**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# G.PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

**(Autonomous)**

Approved by AICTE | NAAC Accreditation with ‘A’ Grade

Accredited by NBA (CSE, ECE & EEE) | Permanently Affiliated to JNTUA

Nandikotkur Road, Venkayapalli (V), Kurnool - 518452, Andhra Pradesh

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

**(Autonomous)**

Approved by AICTE | NAAC Accreditation with ‘A’ Grade

Accredited by NBA (CSE, ECE & EEE) | Permanently Affiliated to JNTUA

Nandikotkur Road, Venkayapalli (V), Kurnool - 518452, Andhra Pradesh



# CERTIFICATE

This is to certify that the project report entitled **“Lung Cancer Detection Using Image Segmentation”** being submitted by **P Pranoy Davis(20at1a05a6) P Pavan Kumar Reddy (20at1a05a2) PD Mahammad Hussain(20at1a0575) L Manjunath Naidu(20at1a0583)**in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering of G. Pullaiah College of Engineering and Technology, Kurnool is a record of bonafide work carried out by them under my guidance and supervision. The results embodied in this project report have not been submitted to any other university or institute for the award of any Degree or Diploma.

|  |  |
| --- | --- |
| Internal Guide | Head of the Department |
| **Mr VIJAYA CHANDRA RAO M.Tech.** | **Mrs. M. SRI LAKSHMI M.Tech.,P.hd** |
| Assistant Professor    Date of Viva-Voce | Associate Professor |
| INTERNAL EXAMINER | EXTERNAL EXAMINER |
|  |  |

## ACKNOWLEDGEMENT

We thank our project supervisor, **Mr VIJAYA CHANDRA RAO M.Tech.** for his guidance, valuable suggestions, and support in the completion of the project.

We would like to express our deep sense of gratitude and our sincere thanks to HOD **Mrs M. SRI LAKSHMI M.Tech.,(Ph.D).** Department of Computer Science and Engineering, G. Pullaiah College of Engineering and Technology, Kurnool for providing the necessary facilities and encouragement towards the project work.

We owe indebtedness to our principal **Dr. C. SREENIVASA RAO M.Tech., Ph.D.** G. Pullaiah College of Engineering and Technology, Kurnool for providing us the required facilities.

We are extremely grateful to Chairman **Mr. G. V. M. MOHAN KUMAR** of G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh for their good blessings. We gratefully acknowledge and express our thanks to the teaching and non-teaching staff of the CSE Department.

### Project Associates

|  |  |
| --- | --- |
| **P PRANOY DAVIS** | **(20AT1A05A6)** |
| **P PAVAN KUMAR REDDY**  **PD MAHAMMAD HUSSAIN** | **(21AT5A05A2)**  **(20AT1A0575)** |
| **L MANJUNATH NAIDU** | **(20AT1A0583)** |

## 

## 

## ABSTRACT

The computer-based process of identifying the boundaries of lung from surrounding thoracic tissue on computed tomographic images, which is called segmentation, is a vital first step in radiologic pulmonary image analysis. Many algorithms and software platforms provide image segmentation routines for quantification of lung abnormalities; however, nearly all of the current image segmentation approaches apply well only if the lungs exhibit minimal or no pathologic conditions. In particular, abnormalities such as pleural effusions and masses often cause inaccurate lung segmentation, which greatly limits the use of image processing methods in clinical and research. In this review, a summary of the current methods for lung segmentation on CT images is provided, with special emphasis on the accuracy and performance of the methods in cases with abnormalities and cases with exemplary pathologic findings. The feasibility of each class and its shortcomings are illustrated with the most common lung abnormalities observed on CT images. The proposed approach expresses a method for segmenting the lung region from lung Computer Tomography (CT) images. This method is proposed to obtain an optimal segmented region

**Keywords**: Lungs Images, K means clustering image segmentation, Marker controlled watershed image segmentation.

### CONTENTS

**Abstract**

**CHAPTER 1 INTRODUCTION**

1.1 Introduction

**CHAPTER 2 LITERATURE REVIEW**

2.1 Literature survey

**CHAPTER 3 PROPOSED METHOD AND EXISTING METHOD**

3.1 Existing System

3.2 Proposed System

3.3 Modules

3.4 Methodology

**CHAPTER 4 SYSTEM REQUIREMENTS**

4.1 Functional requirements

4.2 Non-functional requirements

4.3 Hardware Requirements

4.4 Software Requirements

**CHAPTER 5 APPLICATION DETAILS**

5.1 Details

5.1 UML Diagrams

**CHAPTER 6 FRONTEND TECHNOLOGIES**

6.1 HTML

6.2 CSS

6.3 PYTHON

**CHAPTER 7 BACKEND TECHNOLOGIES**

7.1 Django

7.2 My SQL

7.3 Xampp Server

**CHAPTER 8 SOURCE CODE AND IMPLEMENTATION**

8.1 Source Code

8.2 Implementation

**CHAPTER 9 CONCLUSION**

9.1 Conclusion

**BIBLIOGRAPHY**

## CHAPTER 1

## INTRODUCTION

**1.1 Introduction:**

Lung cancers have been identified as one of the world's most serious causes of death [1]. It is among the most malignant tumors that can affect human wellbeing. Its death rate scores among all tumor deaths, and is also the top killer towards male and female cancer death [2-3]. There have been nearly 1.8 million fresh cases of lung cancer annually (13 percent among all cancers), 1.6 million deaths worldwide (19.4 percent among all cancers). Lung cancer is a proliferation of expanding and developing irregular cells into a tumour. Of the other forms of cancer, the death rate of lung cancer is the greatest. Cigarette smoke induces an approximate 85 percent of cases of lung cancer in males and 75 percent in females. Lung cancer is amongst the most terrible illnesses in the developing countries, with a death rate of 19.4 percent. Lung cancer is among the most dangerous cancer worldwide, with lowest success rate following diagnosis, with a steady rise in casualty count per year [4-6]. Advantages of Fuzzy logic in the earlier predictions will lead to result oriented analysis [5]. Survival of lung cancer as a result of diagnosis is directly related to its progress. Yet individuals have a greater success rate it will be found in the early stages of life. Cancer cells are distributed in blood from the lungs, the lymph fluid that covers the lung tissues. The lymph passes into lymph vessels that discharge through lymph nodes in the lungs and chest region. Examination and treatment of lung disease has become one of the biggest obstacles that humanity faces in recent years. Early tumor diagnosis will reliably promote its survival of vast numbers of life around the world. This paper introduces a method that uses a convolutional neural Network (CNN) to identify the lung tumors as malignant/benign.

**CHAPTER 2**

**LITERATURE REVIEW**

**Deep ensemble learning for automatic identification: Silky Sachar, Anuj Kumar.**

The therapeutic nature of medicinal plants and their ability to heal many diseases raises the need for their automatic identifcation. Diferent parts of plants that help in their identifcation include root, fruit, bark, stem but leaf images have been widely used as they are an abundant source of information and are also easily available. This work explores the branch of Artifcial Intelligence, called deep learning, and proposes an Ensemble learning approach to rapidly detect medicinal plants using the leaf image. The medicinal leaf dataset consists of 30 classes. Transfer learning approach was used to initialize the parameters and pretrain Neural networks namely MobileNetV2, InceptionV3, and ResNet50. These component models were used to extract features from the input images and the softmax layer connected to the Dense Layer was used as the classifer to train the models on the concerned dataset. The obtained accuracies were validated using threefold and fvefold crossvalidation. The Ensemble Deep Learning- Automatic Medicinal Leaf Identifcation (EDL-AMLI) classifer based on the weighted average of the component model outputs was used as the fnal classifer. It was observed that the EDL-AMLI outperformed the state-of-the-art pre-trained models such as MobileNetV2, InceptionV3, and ResNet50 by achieving 99.66% accuracy on the test set and average accuracy of 99.9% using threefold and fvefold cross validation.

**Summary:** Automatic detection of medicinal plants opens new doors for the development of medicines to cure diseases that have not yet been cured by allopathy. It will allow the layman to be aware of the plants growing in their surroundings and make utmost use of them to cure common ailments with no possible side efects. Artifcial Intelligence makes this purpose even more achievable. We proposed an Ensemble of deep learning models to automatically detect medicinal plants. The medicinal leaf images were obtained from a medicinal leaf dataset published in Mendeley. By employing Transfer learning.

**Real-Time Identification of Medicinal Plants using Machine Learning Techniques: Sivaranjani.C, Lekshmi Kalinathan , Amutha.R**

The lighting condition of the environment are uncontrolled, so the segmentation of a leaf from the background is considered as a complex task. Here we propose a system which can identify the plant species based on the input leaf sample. An improved vegetation index, ExG-ExR is used to obtain more vegetative information from the images. The reason here is, it fixes a built-in zero threshold and hence there is no need to use otsu or any threshold value selected by the user. Inspite of the existence of more vegetative information in ExG with otsu method, our ExG-ExR index works well irrespective of the lighting background. Therefore, the ExG-ExR index identifies a binary plant region of interest. The original color pixel of the binary image serves as the mask which isolates leaves as sub-images. The plant species are classified by the color and texture features on each extracted leaf using Logistic Regression classifier with the accuracy of 93.3%.

**Summary:** In this work, we addressed the problem of identifying the medicinal plant species by the analysis of leaf images obtained directly from their habitat and irrespective of lighting conditions. The fixed zero threshold, ExG-ExR vegetative index is successfully tested for image dataset. The result shows that the algorithm can adequately segment the leaf region. This method worked well in images with reflection. The feature extraction based on the color and texture features is done. The classification of medicinal plant species is done by using Weka and the accuracy of 93.3% is measured. In future we have planned to design and develop a system which automatically identifies plant species through the analysis of not only the leaf images also the other parts of the plant acquired directly in their habitat irrespective of complex backgrounds and various lighting condition. R.

**Identification of Medicinal Plants using Deep Learning: R. Upendar Rao, M. Sai Lahari, K. Pavana Sri and team**

Identification of the correct medicinal plants that goes in to the preparation of a medicine is very important in ayurvedic, folk and herbal medicinal industry. The main features required to identify a medicinal plant is its leaf shape, color and texture. Color and texture from both sides of the leaf contain deterministic parameters to identify the species. In this project we explore feature vectors from both the front and back side of a green leaf along with morphological features to arrive at a unique optimum combination of features that maximizes the identification rate. A database of medicinal plant leaves is created from scanned images of front and back side of leaves of commonly used medicinal plants. The leaves are classified based on the shape and dimension combination. It is expected that for the automatic identification of medicinal plants this system will help the community people to develop their knowledge on medicinal plants, help taxonomists to develop more efficient species identification techniques and also participate significantly in the pharmaceutical drug manufacturing

**Summary:** Plants are necessary for human survival. Herbs, particularly, are employed by indigenous populations as folk medicines from old period. Herbs are typically recognized by clinicians based on decades of intimate sensory or olfactory experience. Recent improvements in analytical technology have made it much easier to identify herbs depending on scientific evidence. This helps a lot of individuals, particularly those are not used to recognising herbs. additionally for time-consuming methods, laboratory-based analysis necessitates expertise in sample healing and data explanation. As a result, a simple and reliable method for identifying herbs is required. Herbal identification anticipated to benefit from the combination of computation and statistical examination. This non-destructive technique will be the preferred approach for quickly identifying herbs, especially for individuals who cannot able to use expensive analytical equipment. This work reviews about different methods for plants recognition and also reviews their advantages and disadvantages.

## CHAPTER 3

## EXISTING SYSTEM AND PROPOSED SYSTEM

## 

**3.1 Existing System:**

In the existing there are methods implemented to classify lung disesase classification in deep learning. In method we are performing the classification of lung cancer identification using vgg16 of deep learning along with the Machine learning methods. As image analysis based approaches for classification of medical images.

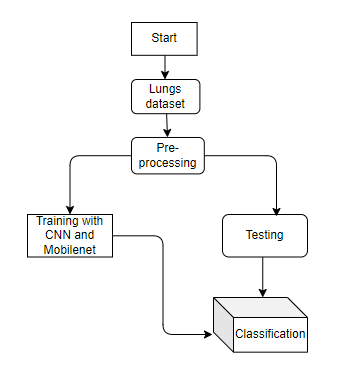
**Disadvantages:**

* Less accuracy.
* More Losses.

**3.2 Proposed system:**

In purposed method we are performing the classification of either the image is lung cancer or normal identification using Mobile net and CNN of deep learning along with the Machine learning methods. As image analysis based approaches for medical lung classification and authentication. Hence, proper classification is important for the lung dataset that which will be possible by using our proposed method. Block diagram of proposed method is shown below.

**Block Diagram:**

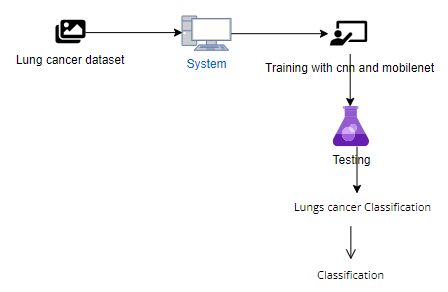


**Fig 1. Block diagram of proposed method**

**Advantages**:

* Accurate classification
* Less complexity
* High performance
* Easy Identification

**ARCHITECTURE**



**3.3 Modules:**

**System**

**User**

**1. System:**

1.1 Create Dataset:

The dataset containing images of the Lung Classification i.e., normal are to be classified is split into training and testing dataset with the test size of 30-20%.

1.2 Pre-processing:

Resizing and reshaping the images into appropriate format to train our model.

1.3 Training:

Use the pre-processed training dataset is used to train our model using Mobile net Deep learning algorithm along with CNN transfer learning methods.

1.4 Classification:

The results of our model are display of lungs images classification.

**2. User:**

2.1 Upload Image

The user has to upload an image which needs to be classified.

2.2 View Results

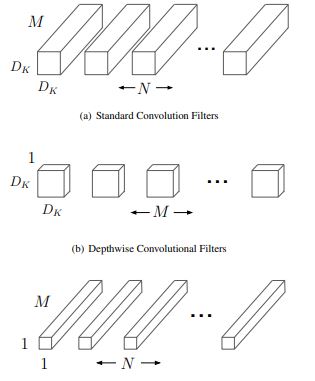
The classified image results are viewed by user.

**3.4 METHODOLOGY:**

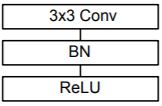
**Mobile Net architecture**.

Convolutional Neural Networks (CNN) have become very popular in computer vision. However, in order to achieve a higher degree of accuracy modern CNNs are becoming deeper and increasingly complex. Such networks cannot be used in real applications like robots and self driving cars.In this section we will be discussing a CNN architecture that aims to effectively tackle this problem.

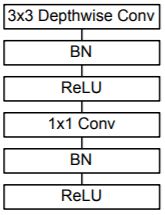
MobileNet is an efficient and portable CNN architecture that is used in real world applications. Mobile Nets primarily use **depth wise separable convolutions** in place of the standard convolutions used in earlier architectures to build lighter models.MobileNets introduce two new global hyper parameters(width multiplier and resolution multiplier) that allow model developers to trade off **latency** or **accuracy** for speed and low size depending on their requirements.



**Standard Convolution layer:**   
A single standard convolution unit (denoted by **Conv** in the table above) looks like this:



**Depth wise separable Convolution layer**  
A single Depth wise separable convolution unit (denoted by **Conv dw** in the table above) looks like this:



**Width**   
Width muliplier (denoted by α) is a global hyperparameter that is used to construct smaller and less computionally expensive models.Its value lies between 0 and 1.For a given layer and value of α, the number of input channels 'M' becomes α \* M and the number of output channels 'N' becomes α \* N hence reducing the cost of computation and size of the model at the cost of performance.The computation cost and number of parameters decrease roughly by a factor of α2.Some commonly used values of α are 1,0.75,0.5,0.25.

**Resolution**  
The second parameter introduced in MobileNets is called resolution multiplier and is denoted by ρ.This hyper parameter is used to decrease the resolution of the input image and this subsequently reduces the input to every layer by the same factor. For a given value of ρ the resolution of the input image becomes 224 \* ρ.This reduces the computational cost by a factor of ρ2.

**CNN**:

A convolutional neural network (CNN or convnet) is a subset of [machine learning](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML). It is one of the various types of artificial [neural networks](https://www.techtarget.com/searchenterpriseai/definition/neural-network) which are used for different applications and data types. A CNN is a kind of network architecture for [deep learning](https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network) algorithms and is specifically used for [image recognition](https://www.techtarget.com/searchenterpriseai/definition/image-recognition) and tasks that involve the processing of [pixel](https://www.techtarget.com/whatis/definition/pixel) data.

There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision ([CV](https://www.techtarget.com/searchenterpriseai/definition/machine-vision-computer-vision)) tasks and for applications where object recognition is vital, such as [self-driving cars](https://www.techtarget.com/searchenterpriseai/definition/driverless-car) and [facial recognition](https://www.techtarget.com/searchenterpriseai/definition/facial-recognition).

### Inside convolutional neural networks

Artificial neural networks (ANNs) are a core element of deep learning algorithms. One type of an ANN is a recurrent neural network ([RNN](https://www.techtarget.com/searchenterpriseai/definition/recurrent-neural-networks)) that uses sequential or time series data as input. It is suitable for applications involving natural language processing ([NLP](https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP)), language translation, [speech recognition](https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition) and image captioning.

The CNN is another type of neural network that can uncover key information in both time series and image data. For this reason, it is highly valuable for image-related tasks, such as image recognition, object classification and [pattern recognition](https://www.techtarget.com/whatis/definition/pattern-recognition). To identify patterns within an image, a CNN leverages principles from linear algebra, such as matrix multiplication. CNNs can also classify audio and [signal](https://www.techtarget.com/searchnetworking/definition/signal) data.

A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, a CNN's neurons are arranged like the brain's frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and also with speech or audio signal inputs.

**SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:**

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.

A diagram of a software development

Description automatically generated

* **Requirement Gathering and analysis** − all possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**FEASIBILITY STUDY**

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

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**Economic feasibility:**

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

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

**Social feasibility:**

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.

**CHAPTER 4**

## SYSTEM REQUIREMENTS

**Functional and non-functional requirements:**

Requirement’s analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

**4.1 Functional Requirements**: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1. Authentication of user whenever he/she logs into the system
2. System shutdown in case of a cyber-attack
3. A verification email is sent to user whenever he/she register for the first time on some software system.

**4.2Non-functional requirements**: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.  
They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability
* Flexibility

Examples of non-functional requirements:

1. Emails should be sent with a latency of no greater than 12 hours from such an activity.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds whenever of simultaneous users are > 10000

**4.3 HARDWARE REQUIREMENTS:**

* Processor : i5/Intel Processor
* Hard Disk : 128 GB
* Ram : 8GB

**4.4 SOFTWARE REQUIREMENTS:**

* Operating System : Windows 10
* Server-side Script : Python 3.6
* IDE : PyCharm, Jupyter notebook
* Libraries Used : Numpy, IO, OS, Django, Keras, pandas, tensorflow

## CHAPTER 5

## APPLICATION DETAILS

### 5.1 Details

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

* Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −
* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding −

What are the inputs needed for the system?

How end users respond to different elements of forms and screens.

### Objectives for Input Design:

The objectives of input design are

To design data entry and input procedures

To reduce input volume

To design source documents for data capture or devise other data capture methods

To design input data records, data entry screens, user interface screens, etc.

To use validation checks and develop effective input controls.

**Output Design:**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design:

The objectives of input design are:

To develop output design that serves the intended purpose and eliminates the production of unwanted output.

To develop the output design that meets the end user’s requirements.

To deliver the appropriate quantity of output.

To form the output in appropriate format and direct it to the right person.

To make the output available on time for making good decisions.

**5.2 UML DIAGRAMS**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling 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 Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling 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.

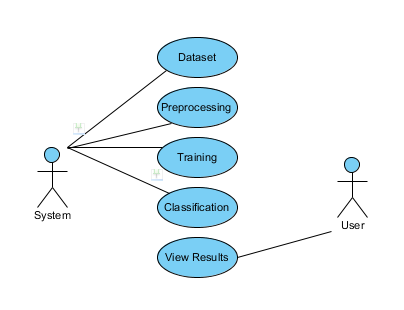
**GOALS:**

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

1. Provide users a ready-to-use, expressive visual modelling 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 modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

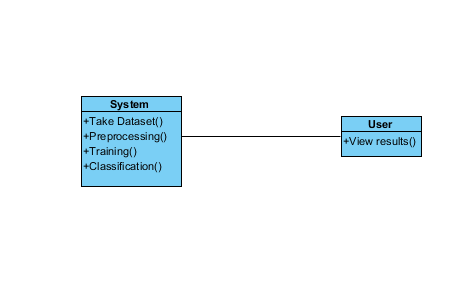
**USE CASE DIAGRAM**

* A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
* Its purpose is to present a graphical overview of the functionality 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.



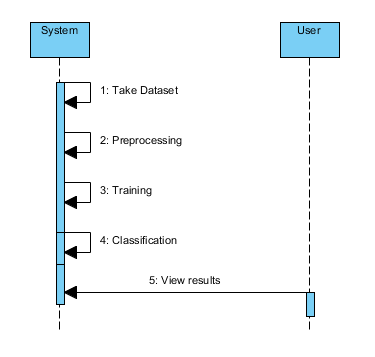
**CLASS DIAGRAM**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



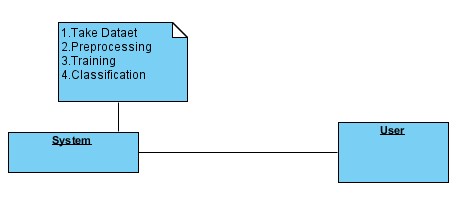
**SEQUENCE DIAGRAM**

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



**COLLABORATION DIAGRAM:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



**DEPLOYMENT DIAGRAM:**

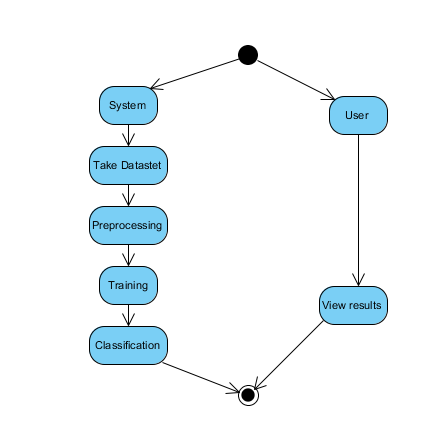
Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.

A blue cube with black lines

Description automatically generated

**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**COMPONENT DIAGRAM**:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.

A black line on a white background

Description automatically generated

**ER DIAGRAM:**

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.

A diagram of a system

Description automatically generated

**DFD DIAGRAM:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

A diagram of a software company

Description automatically generated with medium confidence

A diagram of a data processing process

Description automatically generated

## CHAPTER 6

## FRONTEND TECHNOLOGIES

### 6.1 HTML

HTML (Hypertext Markup Language) is the standard markup language used to create and design web pages. It provides the structure and content of a webpage, defining elements such as headings, paragraphs, links, images, tables, and forms. Here's a brief overview of HTML: **Elements**: HTML documents are composed of HTML elements, which are enclosed in tags and define the structure and content of the webpage. Elements consist of an opening tag, content, and a closing tag. For example:

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

**Attributes**: HTML elements can have attributes that provide additional information about the element, such as its appearance or behavior. Attributes are specified within the opening tag of an element. For example:

<img src="image.jpg" alt="Image Description">

<a href="https://[www.example.com" t](http://www.example.com/)arget="\_blank">Visit Example</a>

**Document Structure**: HTML documents have a hierarchical structure defined by nested elements. The <html> element serves as the root of the document, containing two main **sections**: <head> and <body>. The <head> section contains meta-information about the document, such as the title and links to stylesheets, while the <body> section contains the visible content of the webpage.

**Semantic Elements**: HTML5 introduced semantic elements that provide meaning to the content of a webpage, making it more accessible to both humans and machines. Examples of semantic elements include <header>, <nav>, <main>, <section>, <article>, <footer>, etc. **Links and Navigation**: HTML allows you to create hyperlinks using the <a> (anchor) element, which allows users to navigate between different web pages or sections within the same page. Links can point to other web pages, specific sections within the same page (using anchor links), or external resources like images and documents.

**Images and Multimedia**: HTML provides elements such as <img> for embedding images, <audio> for embedding audio files, and <video> for embedding video files. These elements allow you to include multimedia content within your web pages.

**6.2 CSS(BOOTSTRAP):**

CSS stands for Cascading Style Sheets, and it is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS describes how elements should be displayed on a webpage, including aspects such as layout, colors, fonts, and spacing. Introduction to Bootstrap: Bootstrap is a free and open-source front-end framework developed by Twitter. It provides pre-designed HTML and CSS-based design templates for typography, forms, buttons, navigation, and other interface components, as well as optional JavaScript extensions. Bootstrap is designed to be mobile-first and responsive, meaning that it automatically adjusts the layout and appearance of web pages to fit different screen sizes and devices. Key Features of Bootstrap: Grid System: Bootstrap utilizes a responsive, mobile-first grid system based on a 12-column layout. This grid system allows developers to create responsive layouts that adapt to various screen sizes and resolutions. CSS Components: Bootstrap provides a wide range of pre-designed CSS components, such as buttons, forms, navigation bars, dropdowns, alerts, badges, and more. These components can be easily integrated into web pages to enhance their functionality and appearance. Responsive Design: Bootstrap's responsive design features enable web pages to adapt and scale seamlessly across different devices and screen sizes, including desktops, laptops, tablets, and smartphones. This ensures a consistent user experience across all devices. Customizable Styles: While Bootstrap comes with predefined styles and components, it also offers extensive customization options. Developers can modify and extend Bootstrap's styles using custom CSS or Sass variables to match the design requirements of their projects. Browser Compatibility: Bootstrap is designed to be compatible with modern web browsers, ensuring consistent rendering and functionality across different browsers and browser versions. It includes built- in CSS normalization and support for vendor prefixes to address cross-browser inconsistencies. Community Support: Bootstrap has a large and active community of developers, designers, and contributors who regularly contribute to its development, provide support, and share resources such as themes, templates, and plugins. This community-driven approach fosters collaboration and innovation within the Bootstrap ecosystem. How Bootstrap Works: Integration: To use Bootstrap in a project, developers can include the Bootstrap CSS and JavaScript files in their HTML documents. Bootstrap can be integrated into projects in various ways, including downloading the Bootstrap files and linking them locally, or using Content Delivery Networks (CDNs) to link to hosted Bootstrap files. HTML Structure: Bootstrap components are implemented using HTML markup with predefined CSS classes. Developers can add these classes to HTML elements to apply Bootstrap styles and functionality to them. For example, adding the "btn" class to a button element will style it as a Bootstrap button. Responsive Grid: Bootstrap's grid system allows developers to create responsive layouts by dividing the page into rows and columns. Developers can specify the size of each column and how it should behave on different screen sizes using Bootstrap's predefined grid classes (e.g., col-md-6 for a column that spans half the width on medium-sized screens). Component Customization: Bootstrap components can be customized using CSS or Sass variables to modify colors, fonts, sizes, spacing, and other visual properties. Developers can override Bootstrap's default styles or create their own custom styles to match the design requirements of their projects. JavaScript Plugins: Bootstrap includes optional JavaScript plugins for adding interactive functionality to components such as dropdowns, modals, carousels, tooltips, and more. Developers can enable these plugins by including the Bootstrap JavaScript file in their projects and initializing the plugins using JavaScript code. Advantages of Using Bootstrap: Rapid Development: Bootstrap provides pre-designed CSS and HTML components that can be easily integrated into projects, speeding up the development process and reducing the need for custom styling and scripting. Consistency: Bootstrap ensures consistency in design and layout across web pages, making it easier for users to navigate and interact with websites and web applications. Responsive Design: Bootstrap's responsive grid system and components enable developers to create mobile-first, responsive designs that adapt to different screen sizes and devices, enhancing usability and accessibility. Cross-browser Compatibility: Bootstrap is designed to be compatible with modern web browsers, ensuring consistent rendering and functionality across different browser platforms and versions.

**6.3 PYTHON**

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

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

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

**Python Features**

Python's features include −

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

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

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

**Dynamic vs. Static**

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

**Variables**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

**Standard Data Types**

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

## Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

## Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

## Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

# Python libraries

1. Requests. The most famous http library written by Kenneth remits. It’s a must have for every python developer.

2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.

3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQL Alchemy. A database library. Many love it and many hate it. The choice is yours.

6. Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of mine craft was made

13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUI’s for my python scripts.

14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

15. Scaly. A packet sniffer and analyzer for python made in python.

16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**NumPy**

Humpy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

**Matplotlib**

• High quality plotting library.

**Python class and objects**

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects. When I first learned C and C++, I did great; functions just made sense for me. Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing. One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

**Inheritance**

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by sub classing a class, you can change the behaviour of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

**Operator Overloads**

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

**Exceptions**

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors. They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop. Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

• Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing. Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.User-Defined Exceptions I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.You probably won’t have to do this very often but it’s nice to have the option when necessary.However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.Making your own exceptions involves object-oriented programming, which will be covered in the next chapter. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.To make a custom exception, simply inherit the base exception and define what it will do.

**Python modules**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

As indicated above, code is usually developed in a file using an editor.To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. This entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

## There are three types of functions in python:

## Help (), min (), print ().

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* global names of a module
* local names in a function or method invocation
* built-in names: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

**Garbage Collection**

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

## CHAPTER 7

## BACKEND TECHNOLOGIES

**7.1 Django**

Django is a high-level web framework written in Python that encourages rapid development and clean, pragmatic design. It follows the Model-View-Controller (MVC) architectural pattern, although in Django this pattern is commonly referred to as Model-View-Template (MVT).

Here's an overview of Django's key features and components:

* Object-Relational Mapping (ORM)
* Admin Interface
* URL Routing
* View Layer
* Template Engine
* Forms
* Authentication and Authorization
* Security Features
* Internationalization and Localization
* Testing Framework
* REST Framework

**7.2 MYSQL**

MySQL is an open-source relational database management system (RDBMS) that is widely used for storing and managing structured data. It is known for its reliability, performance, and ease of use, making it a popular choice for web applications, data warehousing, and other data-driven projects. Here's an overview of MySQL's key features and components:

Relational Database SQL (Structured Query Language)

* Client-Server Architecture
* Storage Engines
* Transactions and ACID Compliance
* Indexes and Optimization
* Replication and High Availability
* Security Features
* Backup and Recovery

**7.3 XAMPP Server**

XAMPP is an open-source cross-platform web server solution stack package developed by Apache Friends. It consists mainly of the Apache HTTP Server, MariaDB database (MySQL database until XAMPP version 5.5.30), and interpreters for scripts written in the PHP and Perl programming languages.

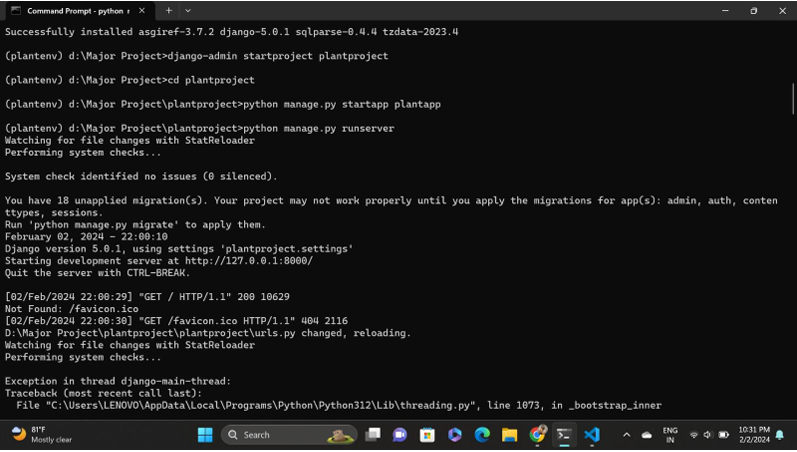
Here's a breakdown of the components included in XAMPP:

* Apache HTTP Server
* MariaDB Database (formerly MySQL)
* PHP Interpreter
* Perl Interpreter
* phpMyAdmin
* FileZilla FTP Server (optional)

A screenshot of a computer

Description automatically generated

## Creating virtual environment

Django installation

A screenshot of a computer

Description automatically generated

Registration database

## CHAPTER 8

## SOURCE CODE AND IMPLEMENTATION

### 8.1 Source code

from django.shortcuts import render

# Create your views here.

#Importing Libraries required for the code

from django.shortcuts import render

from .models import Monument

import numpy as np

import os

from tensorflow.keras.preprocessing import image

from tensorflow.keras.models import load\_model

# Create your views here.

#Creating variables

indexhtml='index.html'

abouthtml='about.html'

uploadhtml='upload.html'

resulthtml='result.html'

#Rendering Pages

def home(request):

    return render(request,indexhtml)

def about(request):

    return render(request,abouthtml)

#Loading the Model

#Saving the image file uploaded

#Using the saved file to predict

#Assigining the message to predictions

def upload(request):

    pathss=os.listdir("app/dataset/test")

    classes=[]

    for i in pathss:

        classes.append(i)

    if request.method=='POST':

        m1 = int(request.POST['alg'])

        file=request.FILES['data']

        img=Monument(image=file)

        img.save()

        if m1==1:

            path="app/static/saved/"+ img.filename()

            path1="/static/saved/"+ img.filename()

            models=load\_model("app/models/CNN\_1.h5")

        elif m1==2:

            path="app/static/saved/"+ img.filename()

            path1="/static/saved/"+ img.filename()

            models=load\_model("app/models/mobilenet.h5")

        x=image.load\_img(path,target\_size=(224,224))

        x=image.img\_to\_array(x)

        x=np.expand\_dims(x,axis=0)

        x/=255

        results=models.predict(x)

        b=np.argmax(results)

        prediction=classes[b]

        return render(request,resulthtml,{'result':prediction,'path':path1})

    return render(request,uploadhtml)

### 8.2 Implementation

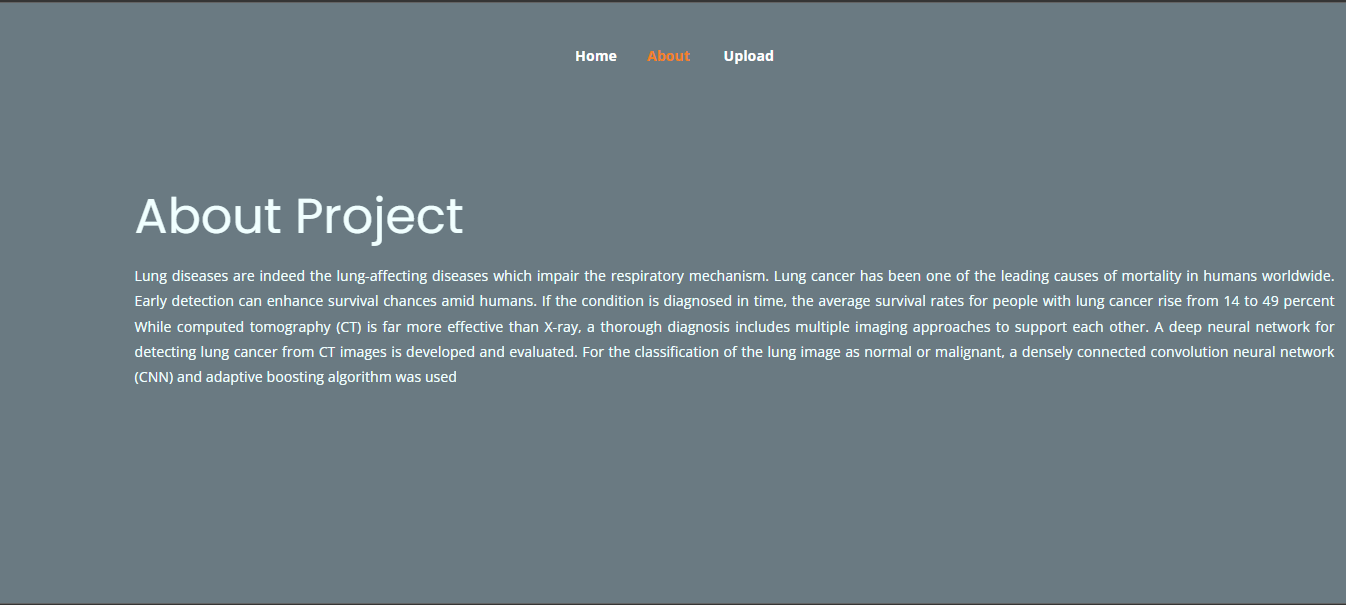
**OUTPUT SCREEN SHOTS WITH DESCRIPTION.** A screenshot of a computer

Description automatically generated

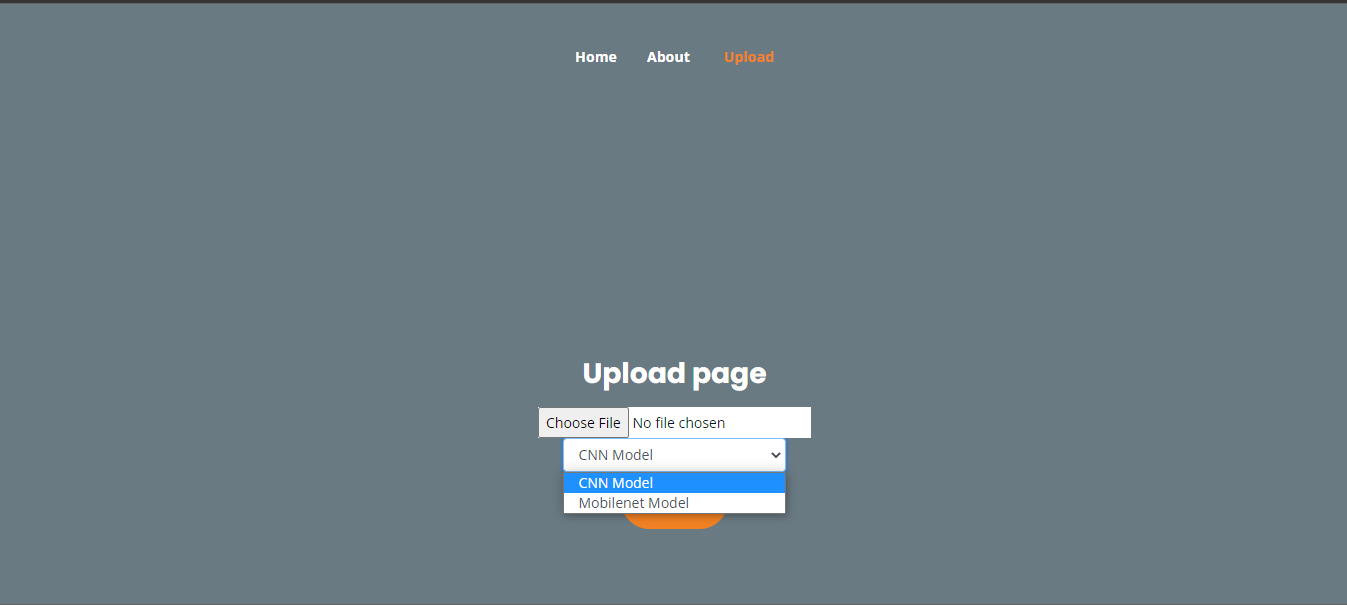
**Home:** In our project, we are classifying the LungImage Classification, with the help of deep learning and Transfer learning.

**About page:**

Small description about project



**Upload Image with model selection:** Here the images can be uploaded those which are to be classified.



**Classified output:** The uploaded image is classified.



**TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Result** |
| Input text | Tested for the classification of Lung Cancer | Success |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **Test cases** | **I/O** | **Expected O/T** | **Actual O/T** | **P/F** |
| 1 | Read the dataset. | Dataset path. | Dataset need to read successfully. | Dataset fetched successfully. | P |
| 2 | Performing pre-processing on the dataset | Pre-processing part takes place | Pre-processing should be performed on dataset | Pre-processing successfully completed. | P |
| 3 | Model Building | Model Building for the clean data | Need to create model using required algorithms | Model Created Successfully. | P |
| 4 | Classification | Input image provided. | Output should be Lung Classification. | Model classified successfully | P |

**TEST CASES MODEL BUILDING:**

## CHAPTER 9

## CONCLUSION

### Conclusion:

In this project we have successfully classified the images of Identification of lungs and liver disease, or its healthy using the deep learning and machine learning. Here, we have considered the dataset of lungs images which will be of different types (healthy or disease) and trained using Mobile Net, CNN along with some transfer learning method. After the training we have tested by uploading the image and classified it.Future scope is that this can be utilized in future to classify the types of different lung cancer easily that which can tend to easy Prediction of diseases in early stages and can take the initial medications, precautions and take measures.

## BIBLIOGRAPHY

[1]. Parekh J, Chanda S (2010) Antibacterial and phytochemical studies on twelve species of Indian medicinal plants. Afr J Biomed Res 10:175–181. <https://doi.org/10.4314/ajbr.v10i2.50624>

[2]. Benarba B, Pandiella A (2018) Colorectal cancer and medicinal plants: principle fndings from recent studies. Biomed Pharmacother 107:408–423. <https://doi.org/10.1016/j.biopha.2018.08.006>

[3]. Adhikari B, Marasini BP, Rayamajhee B et al (2021) Potential roles of medicinal plants for the treatment of viral diseases focusing on COVID-19: a review. Phyther Res 35:1298–1312. https:// doi.org/10.1002/ptr.6893

[4]. de Freitas Junior LM, de Almeida EB (2017) Medicinal plants for the treatment of obesity: ethnopharmacological approach and chemical and biological studies. Am J Transl Res 9:2050–2064

[5]. Wäldchen J, Mäder P (2018) Plant species identifcation using computer vision techniques: a systematic literature review. Arch Comput Methods Eng 25:507–543. https://doi.org/10.1007/ s11831-016-9206-z

[6]. Sachar S, Kumar A (2020) Survey of feature extraction and classifcation techniques to identify plant through leaves. Expert Syst Appl. <https://doi.org/10.1016/j.eswa.2020.114181>

[7]. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22:1345–1359. [https://doi.org/10.1109/TKDE. 2009.191](https://doi.org/10.1109/TKDE.%202009.191)

[8]. Lee SH, Chan CS, Mayo SJ, Remagnino P (2017) How deep learning extracts and learns leaf features for plant classifcation. Pattern Recognit 71:1–13. <https://doi.org/10.1016/j.patcog.2017.05.015>

[9]. Grinblat GL, Uzal LC, Larese MG, Granitto PM (2016) Deep learning for plant identifcation using vein morphological patterns. Comput Electron Agric 127:418–424. [https://doi.org/10.1016/j. compag.2016.07.003](https://doi.org/10.1016/j.%20compag.2016.07.003)

[10]. Gogul I, Kumar VS (2017) Flower species recognition system using convolution neural networks and transfer learning. 2017 4th Int Conf Signal Process Commun Networking. ICSCN 2017:1–6. <https://doi.org/10.1109/ICSCN.2017.8085675>

[11]. Liu Z, Zhu L, Zhang XP et al (2015) Hybrid deep learning for plant leaves classifcation. Lect Notes Comput Sci (Include Subser Lect Notes Artif Intell Lect Notes Bioinform) 9226:115–123. https://doi.org/10.1007/978-3-319-22186-1\_11