# MULTI-CANCER DETECTION SYSTEM- A DEEP LEARNING APPROACH

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**

**Batch – A15**

**K. Pranathi**
(21JG1A0558)

**I. Kushyanthi**
(21JG1A0537)

**G. Sai Kumari**
(21JG1A0530)

**B. Meghana S.S.L**
(21JG1A0565)

**Under the esteemed guidance of**

**Dr. P.V.S. Lakshmi Jagadamba**

Professor & Head

CSE Department

**Department of Computer Science and Engineering**

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN**

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation (NBA) for B. Tech. CSE, ECE & IT – valid from 2019-22 and 2022-25]

[Accredited by National Assessment and Accreditation Council (NAAC) with A Grade-Valid from 2022-2027] Kommadi, Madhurawada, Visakhapatnam – 530048

**2021 – 2025**

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## CERTIFICATE

This is to certify that the Project report titled "**MULTI-CANCER DETECTION SYSTEM - A DEEP LEARNING APPROACH"** is a bonafide work of following IV B.Tech II sem students in the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2024-25, in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology of this University.

**K. Pranathi** (21JG1A0558)
**G. Sai Kumari** (21JG1A530)
**I. Kushyanthi** (21JG1A0537)
**B. Meghana S.S.L**(21JG1A0565)

Signature of the guide
**Dr. P. V. S. Lakshmi Jagadamba**
Professor
Dept. of CSE

Signature of the HOD
**Dr. P. V. S. Lakshmi. Jagadamba**
Professor
Dept. of CSE

**External Examiner**

# ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We feel elated to extend our sincere gratitude to our guide, **Dr. P. V. S. Lakshmi Jagadamba**, Professor & Head of the Department of Computer Science and Engineering for encouragement all the way during analysis of the project and for expressing her valuable and grateful opinions in the project for its development. Her annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We would like to take this opportunity to extend our gratitude to Project Coordinator, **Dr. V. Lakshman Rao,** Associate Professor of Computer Science and Engineering for making us to follow a defined path and for advising us to get better improvements in the project.

We would like to take this opportunity to express our profound sense of gratitude to **Dr. R. K. Goswami,** Principal and **Dr. G. Sudheer,** Vice Principal for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

We are also thankful to both teaching and non-teaching faculty of the Department of Computer Science and Engineering for giving valuable suggestions for our project.

**K. Pranathi** (21JG1A0558)

**I. Kushyanthi** (21JG1A0537)

**G. Sai Kumari**(21JG1A0530)

**B. Meghana S. S. L**(21JG1A0565)

# TABLE OF CONTENTS

# ABSTRACT

Early and accurate diagnosis of cancer is important to enhance patient survival and facilitate on-time treatment planning. The proposed work seeks to build an AI-based Multi-Cancer Detection System to classify brain, lung, and breast cancers as cancerous or non-cancerous from medical imaging data. The system employs a pre-trained ResNet50 convolutional neural network to perform deep feature extraction and classification. To counter class imbalance and enhance learning of challenging instances, Focal Loss is utilized in training. A complete preprocessing pipeline consisting of denoising, contrast enhancement with CLAHE, and brightness correction is used to provide high-quality, normalized input images. Further, a scan validator is also incorporated to remove non-medical images, upholding prediction integrity. The model had a 95% overall classification accuracy over six cancer classes, indicating its high precision and stability in clinical diagnostic applications. This paper provides a trustworthy, AI-based system that facilitates early cancer detection and assists healthcare providers in precise and effective clinical decision-making.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SCREENS

# 1. INTRODUCTION

In medical diagnostics, early and correct detection of cancer is key in enhancing patient care and saving lives. Brain, lung, and breast cancer are some of the most common and lethal types of the disease globally. Cancer detection has conventionally depended on trained radiologists visually interpreting medical imaging data like MRI, CT, and mammogram scans. However, this manual process is time-consuming, prone to human error, and limited by inter-observer variability. In many cases, subtle signs of cancer may be missed in the early stages, leading to delayed treatment and reduced survival rates. To tackle these issues, this project proposes a deep learning–based Multi-Cancer Detection System capable of automatically diagnosing medical images as cancerous or non-cancerous across various cancer types—namely brain, lung, and breast cancers. In contrast to single-use diagnostic equipment that aims to detect a single type of cancer, this system exploits the potential of Convolutional Neural Networks (CNNs) to generalize across various imaging modalities and anatomical structures. The system to be proposed processes the input images via a stable image preprocessing pipeline, followed by a ResNet50 architecture that is fine-tuned, and extracts deep features for precise classification. The model is also augmented with Focal Loss for dealing with class imbalances usually present in medical data. Predictions also come with confidence scores providing transparency and interpretability to doctors. Initial automated cancer detection approaches relied on traditional machine learning methods like Decision Trees and Support Vector Machines. This multi-cancer early detection system can be applied to wide-ranging clinical decision support, telemedicine, and screening initiatives, with the potential to lower diagnostic errors, expand access in underserved populations, and aid physicians in quicker and more knowledgeable decisions.

## 1.1 PROBLEM DEFINITION

The primary problem addressed by this project is the lack of an efficient, automated system for detecting multiple types of cancer, particularly brain, lung, and breast cancers. Current diagnostic methods are often time-consuming, expensive, and dependent on expert analysis, which can delay treatment and reduce accuracy. Moreover, many existing systems are limited to identifying a single cancer type, limiting their applicability in real-world healthcare environments. This limitation creates a need for a more versatile and scalable solution. The challenge lies in building a deep learning-based system that can analyze medical images and accurately classify multiple cancer types in real-time. Such a system would support early diagnosis, assist medical professionals, and ultimately improve patient care and outcomes.

## 1.2 OBJECTIVE OF THE PROJECT

The objective of this project is to develop and implement an automated deep learning-based system capable of detecting and classifying multiple types of cancers specifically lung, brain, and breast cancers using CT and MRI medical images. The system utilizes a ResNet-50 convolutional neural network architecture to accurately differentiate between cancerous and non-cancerous cases across these three cancer types. To address the challenge of class imbalance, methods such as Focal Loss and data augmentation are employed. By achieving these objectives, the project aims to support early cancer detection, assist medical professionals in diagnosis, and ultimately contribute to improving patient outcomes and healthcare efficiency through AI-driven diagnostic support.

## 1.3 LIMITATIONS OF THE PROJECT

Despite its potential, the multi-cancer detection system has some limitations. The model's accuracy can be affected by variations in image quality, resolution, and scanning protocols. Class imbalance may lead to biased predictions, particularly for less-represented classes. The system is limited to classification and does not offer tumor localization or segmentation. Additionally, real-time deployment may be restricted by hardware requirements, making it less suitable for low-resource environments.

## 1.4 ORGANIZATION OF THE DOCUMENTATION

Concise overview of rest of the documentation work is explained below:

**Chapter 1:** Introduction describes the motivation and objective of this project.
**Chapter 2:** Literature survey describes the primary terms involved in the development of the project.

**Chapter 3:** Analysis deals with the detailed analysis of the project. Software requirement specification further contains software requirements, hardware requirements.

**Chapter 4:** Contains design part and UML diagrams.

**Chapter 5:** Contains step by step process and screenshots of output

**Chapter 6:** Contains conclusion of the project.

**Chapter 7:** Contains future scope of the project.

**Chapter 8:** Contains references.

# 2. LITERATURE SURVEY

Multi-Cancer Detection (MCD) is the identification of various forms of cancers—brain, lung, and breast cancer—using medical images. It facilitates early detection and enhanced treatment outcomes. Traditional methods for detecting cancer were based on handcrafted features and classical machine learning approaches, but now deep learning-based models such as CNNs (e.g., ResNet50) are popularly used for their accuracy and capability to learn intricate patterns from data. In this project, individual cancer detection methods have been learned from different research papers—some on breast cancer, some on brain or lung cancer—and integrated into a single multi-cancer detection system. This integration assists in creating a more comprehensive and intelligent diagnostic tool.

In [1], Dr. R. Vijaya Kumar Reddy et al. proposed "Enhancing Brain Tumor Detection with ResNet," which focuses on classifying brain tumors from MRI scans using deep learning. Using a dataset of 3064 slices from 233 patients, the study applied preprocessing techniques like resizing, skull stripping, and normalization. The ResNet model, with its residual connections, enabled efficient training and achieved a high accuracy of 99.3%. The results highlight the model's potential in clinical applications and provide a foundation for multi-cancer detection systems.

In [2], S. Khawaldeh, M. T. Al-Qaralleh, and A. Al-Rawy proposed "Brain Tumor Detection Based on Deep Learning Approaches and Magnetic Resonance Imaging," integrating ResNet50 for classification, K-means++ for segmentation, and SGLDM for extracting texture features. Synthetic data augmentation improved robustness, while Grad-CAM enhanced model interpretability. Using the Br35H:BrainTumorDetection2020 dataset, the model achieved improved accuracy, sensitivity, and specificity, showing its reliability for MRI-based brain tumor detection.

In [3], O. O. Oladimeji and A. O. J. Ibitoye proposed "Brain tumor classification using ResNet50-convolutional block attention module," using a ResNet50-CBAM model on a dataset of 7,023 MRI images across four classes—glioma, meningioma, pituitary, and no tumor. Images were preprocessed using resizing, normalization, and Dynamic Histogram Equalization. ResNet50 with frozen layers extracted deep features, while CBAM enhanced relevant spatial and channel information. Final classification was performed using global average pooling and SoftMax, improving accuracy in multi-class brain tumor detection.

In [4], P. Sarah, S. Krishnapriya, S. Saladi, Y. Karuna, and D. P. Bavirisetti proposed “A novel approach to brain tumor detection using K-Means++, SGLDM, ResNet50, and synthetic data augmentation,” where segmentation and deep learning were combined for efficient classification. Using 802 MRI images from the Br35H dataset, preprocessing steps included resizing, normalization, and Gaussian blurring. Data augmentation expanded the dataset to 1,604 images with brightness, translation, and rotation adjustments. K-means++ was used for precise tumor segmentation, and classification was performed using ResNet50, showing strong generalization and performance.

In [5], R. A. Patil et al. presented “Deep Learning for Improved Breast Cancer Detection: ResNet-50 vs VGG16,” where high-resolution mammograms from DDSM and TCIA were used to detect benign and malignant cases. Preprocessing included normalization, contrast enhancement, and resizing. ResNet-50 and VGG16 were employed with multiview analysis to boost classification. Data augmentation, regularization, and cross-validation were applied for robust training. Evaluation metrics like accuracy, F1-score, AUC-ROC, and confusion matrix validated model performance, while computational efficiency was analyzed for real-world deployment.

In [6], E. Banjarnahor, J. J. Pangaribuan, O. P. Barus, and R. Romindo presented “Deep Learning in Image Classification Using Modified ResNet50 and Texture Features for Lung Cancer Detection,” where CT scans were processed through contrast adjustment, resizing, normalization, and noise removal. Lung segmentation isolated the region of interest, and nodule detection was based on pixel intensity thresholds. Modified ResNet-50, known for its residual architecture, was used for classifying benign and malignant nodules. The system showed reliable performance in lung cancer detection by combining texture features and deep learning.

In [7], Rashmi Gudur, Nitin Patil, and S.T. Thorat proposed “Early Detection of Breast Cancer using Deep Learning in Mammograms,” combining deep learning with medical imaging for improved diagnostic accuracy. Mammogram images and metadata were preprocessed, and critical features were extracted using a Convolutional Neural Network (CNN). The ResNet50 architecture, with its deep residual layers, detected subtle anomalies that could indicate cancer. A Recurrent Neural Network (RNN) integrated temporal data like patient history to enhance classification, providing a comprehensive risk assessment for early breast cancer detection.

In [8], Sneha Khoriaa and Raghuraj Singh proposed “Predicting Lung Cancer Using ResNet-50 Deep Residual Learning Model using Relu-Memristor Activation Function,” presenting an automated lung cancer detection system using CT scan images. The system integrates image preprocessing (noise removal, contrast enhancement, resizing) with a six-phase lung segmentation method. The ResNet-50 model, enhanced with ReLU and Relu-memristor-like activation functions (RMAF), was used for the final classification of lung nodules. The system showed effective detection of benign and malignant nodules, offering a robust and accurate solution for early lung cancer diagnosis.

In [9], Balamurugan A.G, Saravanan Srinivasan, Preethi D, Monica P, and Sandeep Kumar Mathivanan proposed “Predicting Lung Cancer Using ResNet-50 Deep Residual Learning Model using Relu-Memristor Activation Function,” utilizing a dataset of 7,023 brain MRI images classified into four categories: glioma, meningioma, pituitary tumor, and no tumor. The images were preprocessed using techniques like Fuzzy Dynamic Histogram Equalization (FDHE) and resized to 256x256 pixels to enhance contrast and mitigate overfitting. The proposed ResNet101-CWAM fusion model, integrated with channel-wise attention mechanisms, achieved high accuracy in classifying brain tumors. This study highlights the importance of advanced preprocessing and attention-based CNN models for improving diagnostic accuracy in medical imaging.

In [10] Md. Mahfuz Ahmed et al. proposed “Brain tumor detection and classification in MRI using hybrid ViT and GRU model with explainable AI in Southern Bangladesh," utilizing a novel dataset, *Brain Tumor MRI Hospital Data 2023 (BrTMHD-2023)*. The dataset, consisting of 1,166 MRI scans, was preprocessed with steps like image resizing, normalization, and augmentation, expanding the data to 2,190 images. The study used a hybrid ViT and GRU model for improved brain tumor classification and explainability.

In [11] F. Rustom et al. proposed "Deep learning and transfer learning for brain tumor detection and classification," where a CNN framework was used for glioma classification (astrocytoma, oligodendroglioma, mixed gliomas) using T1-weighted and T2-weighted MRI scans. The dataset consisted of 264 glioma MRIs and 120 normal MRIs, with minimal preprocessing. The model, based on AlexNet, was pretrained on a large dataset and transfer trained on the medical data to reduce overfitting and improve generalization, given the limited annotated MRI data.

In [12], Dr. T. Sunil Kumar et al. proposed "Breast Cancer Classification and Predicting Class Labels Using ResNet50," using a modified ResNet-50 model to classify mammogram images into four cancer types. The CBIS-DDSM dataset was used, with preprocessing techniques like median filtering, Otsu's thresholding, and augmentation to enhance image quality and model performance. An attention mechanism and local max pooling improved feature extraction, and the model was trained using TensorFlow and Keras, achieving robust results in automated breast cancer classification.

In [13], A. F. A. Alshamrani and F. S. Z. Alshomrani proposed "Optimizing breast cancer mammogram classification through a dual approach," using ResNet50 for feature extraction on the KAU-BCMD dataset. The study employed preprocessing techniques like normalization and label encoding, and addressed class imbalance with SMOTE. A custom fully connected model with dropout, L2 regularization, and batch normalization was used for classification into BI-RADS categories. The model was trained with the Adam optimizer and early stopping, achieving robust performance despite data imbalance.

In [14], G. A. Sandag and D. T. Kabo proposed a "Comparative analysis of lung cancer classification models” using ResNet and EfficientNet on CT-scan images. Preprocessing included resizing, augmentation, and contrast enhancement. Transfer learning was applied for classifying adenocarcinoma, squamous, normal, and large cell types. The system, built with Flask, allows users to upload CT images and view results. Performance was assessed using accuracy, precision, recall, and F1-score.

In [15], V. Kumar et al. proposed a “Unified deep learning model for enhanced lung cancer prediction” using ResNet-50, ResNet-101, and EfficientNet-B3 on DICOM images. The CNN architecture includes convolutional, pooling, and fully connected layers. Image features are extracted via sliding K×K kernels, followed by dimensionality reduction using pooling, and classification via FC layers .

Table 1. Literature Survey

| S.No | Title & Authors | Published Journal & Year | Dataset Used | Approach | Outcome | Limitations |
|---|---|---|---|---|---|---|
| 1 | "Enhancing Brain Tumor Detection with ResNet: A Deep Learning Approach" – Dr. R. Vijaya Kumar Reddy | IEEE, Feb 2024 | Brain tumor dataset from Jun Cheng via Figshare (3064 MRI images) | Pre-trained ResNet for classifying brain tumor types | 99.3% accuracy | Only tested on a single dataset; lacks generalization |
| 2 | "Brain Tumor Detection Based on Deep Learning Approaches and MRI" – S. Khawaldeh et al. | ICCAIS | Br35H: BrainTumorDetection 2020 | ResNet50 and other models | Higher accuracy, sensitivity, and specificity | Limited generalizability |
| 3 | "Brain Tumor Classification using ResNet50-CBAM" – O. O. Oladimeji, A. O. J. Ibitoye | Applied Computing and Informatics, 2023 | 7,023 MRI images | ResNet50 + CBAM + DHE preprocessing | Improved accuracy through attention mechanism | No external validation |
| 4 | "Novel Brain Tumor Detection Using K-Means++, SGLDM, ResNet50"– P. Sarah | Frontiers in Physiology, 2024 | 802 MRI images from Br35H | K-Means++ clustering + augmentation | Enhanced segmentation and generalization | No classification; lacks tumor type distinction |
| 5 | "ResNet-50 vs VGG16 for Breast Cancer Detection" – Rupali A. Patil et al. | Int. J. of Electronics & Computer Applications, 2024 | DDSM Mammography Dataset | Transfer learning using ResNet-50 and VGG16 | 96% accuracy, ResNet-50 outperformed VGG16 | No comparison with newer models |
| 6 | "Modified ResNet50 and Texture Features for Lung Cancer Detection" – Evander Banjarnahor | IEEE, 2024 | CT scan images (unspecified) | White pixel + threshold + ResNet50 | Efficient benign/malignant detection | Thresholds may yield false results |
| 7 | "Early Detection of Breast Cancer using DL in Mammograms" – Rashmi Gudur et al. | Journal of Pioneering Medical Sciences, 2024 | Mammograms with metadata | ResNet50 + RNN (spatial-temporal) | High diagnostic accuracy | High computational cost |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | “Predicting Lung Cancer Using ResNet-50 & Relu-Memristor” – Sneha Khoriaa et al. | Engineer Journal, Mar 2025 | CT scan images | ResNet-50 with Relu-Memristor activation | High accuracy, surpasses previous models | Computational complexity; generalization issues |
| 9 | “Robust Brain Tumor Classification Using CWAM” – Balamurugan A.G et al. | BMC Medical Imaging, 2024 | MRI images (4 classes) | ResNet101 + CWAM attention | 99.83% accuracy, 99.27% F1-score | High complexity; dataset limitation |
| 10 | “Hybrid ViT-GRU Model for Brain Tumor with Explainable AI”– Md. Mahfuz Ahmed et al. | Scientific Reports (Nature), 2024 | BSMMCH, BrTMHD-2023, and Kaggle datasets | ViT-GRU + SHAP, LIME, Attention Map | 98.97% accuracy, high interpretability | Complex data, class imbalance, XAI challenges |
| 11 | “DL and Transfer Learning for Brain Tumor Detection” – Faris Rustom et al. | Int. J. of Health Sciences, 2022 | Kaggle Brain Tumor Dataset (3 classes) | VGG16, ResNet50, MobileNetV2 | MobileNetV2: 98.3% accuracy | Pretrained models lack domain-specific tuning |
| 12 | “Breast Cancer Classification Using ResNet50” – T. S. Kumar et al. | Journal of Electrical Systems, 2023 | CBIS-DDSM | ResNet50 + image preprocessing (Otsu thresholding) | 90.6% accuracy | False positives and dataset quality concerns |
| 13 | “Optimizing Mammogram Classification Using Dual Approach”– Abdullah Fahad et al. | IEEE, 2025 | KAU-BCMD Dataset | SMOTE + visualization-based DL framework | 99% accuracy on balanced data | Requires high computational resources |
| 14 | “Comparative Analysis Using EfficientNet & ResNet” – Green Arther Sandag et al. | COGITO Smart Journal, Jun 2024 | 1,000 lung CT scans | EfficientNetB1–B7, ResNet50/101 | EfficientNetB3: 97.78% accuracy | Limited dataset; no real-time |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  |  |  |  |  |  | integration |
| 15 | “Fusion Models for Lung Cancer Prediction” – Vinod Kumar et al. | BMC Medical Imaging, 2024 | 1,000 DICOM images from LIDC-IDRI | ResNet-50, 101, EfficientNet-B3, fusion models | Fusion: 100% precision for Squamous | Overfitting risk; limited dataset diversity |

## 2.1 EXISTING SYSTEM

Based on our literature survey ,there are existing system that proposed for brain tumor diagnosis is based on MRI scans classified as per the presence of the tumor. These scans are subjected to preprocessing operations including image segmentation, quality improvement, resizing (224×224 pixels), normalization of pixel value, and removal of non-relevant features by skull stripping. These operations improve the quality of MRI scans and improve the performance of deep learning models For tumor classification and detection, the system utilizes ResNet, which is a deep learning-based Convolutional Neural Network (CNN). ResNet is strong in feature extraction and hence can be used to detect tumors from complex medical images. Along with classification, segmentation techniques are applied to segment tumor regions. They are as follows:

- Manual Segmentation: Tumors are annotated by expert radiologists.
- Semi-Automatic Segmentation: Region growing and graph cuts algorithms are used to aid the identification of tumors.
- Fully Automatic Segmentation: Deep models such as U-Net and DeepLabV3+ undertake pixel-wise classification to identify the tumor boundaries.

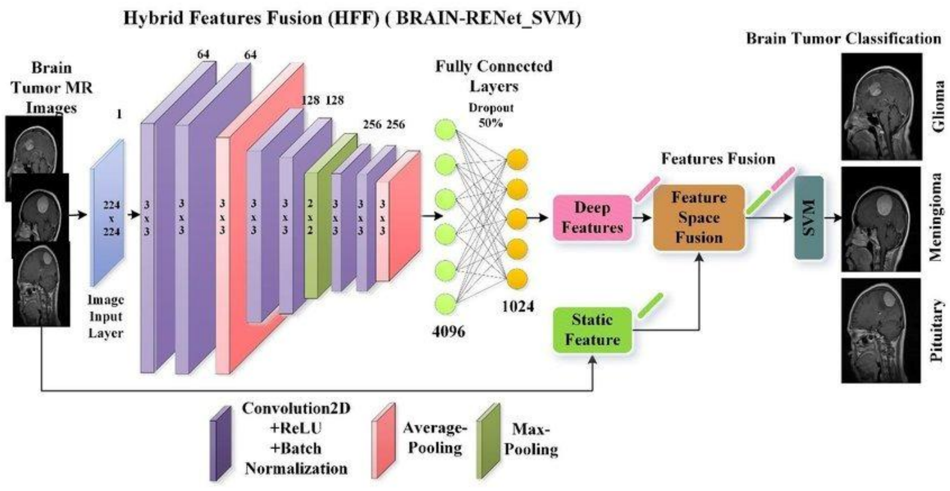


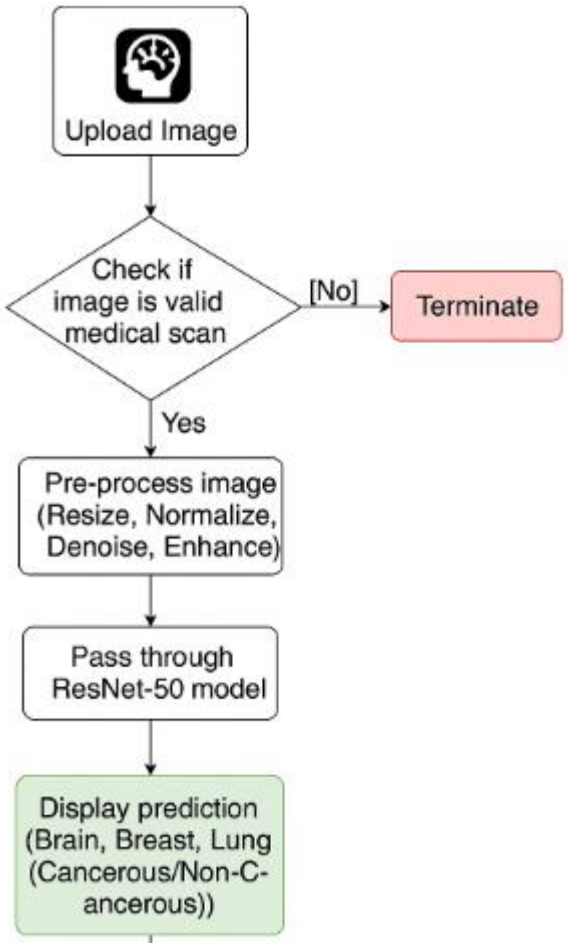
**Fig.1: Existing System**

## 2.2 LIMITATION OF EXISTING SYSTEM:

- Limited Dataset & Generalization Issues – A lack of diverse, high-quality MRI images affects the model's ability to generalize, leading to potential biases in tumor detection.
- Complex Tumor Differentiation – Variations in size, shape, and intensity make it challenging to distinguish tumors from healthy tissue, impacting classification accuracy.
- High Computational Requirements & Misclassification Risks – Deep learning models like ResNet require significant GPU power and large datasets, while false positives and negatives can lead to misdiagnosis.

## 2.3 PROPOSED SYSTEM

The project proposed utilizes deep learning models (ResNet-50 and ResNet-101) for accurate detection and classification of brain, lung, and breast cancer based on medical imaging data. It processes a dataset comprising 7,000+ brain tumor images, 1,000+ lung cancer images, and 2,000+ breast cancer images, categorizing them based on tumor type and severity. To enhance image quality and model performance, preprocessing techniques such as image normalization, resizing (224×224 pixels), color space conversion (BGR to RGB), and data augmentation (including rotation, flipping, and brightness adjustment) are applied. The system employs an ensemble model (ResNet-50) to improve classification accuracy and generalization. A web interface allows users to upload medical images, which are then preprocessed and analyzed in real time to predict the presence of cancer. The system is a valuable tool for assisting healthcare professionals in early cancer detection and diagnosis.



Upload Image
Check if image is valid medical scan
[No]
Terminate
Yes
Pre-process image (Resize, Normalize, Denoise, Enhance)
Pass through ResNet-50 model
Display prediction (Brain, Breast, Lung (Cancerous/Non-C-ancerous))

**Fig.2: Proposed System Architecture**

# 3. REQUIREMENT ANALYSIS

## 3.1 INTRODUCTION

The initial part of software development is requirement analysis. The primary goal of this phase is to define the problem and the system that will be developed. The later stages are completely dependent on this phase, hence the system analyst must be more specific and precise about it. Any irregularity in this step will cause complications in the subsequent phases. The main activities include:

- Software Requirements Specification
- Creating Content diagram
- Flowchart of the code

## 3.2 SOFTWARE REQUIREMENT SPECIFICATION

### 3.2.1 Software Requirements

The software requirements include the languages, libraries, packages and operating system, and different tools for developing the project. We used python for coding Refer table 1 and table 2 for software requirements

#### 3.2.1.1 Minimum Software Requirements

**Table 2: Software Requirements**

| **Software** | **Version** |
|---|---|
| Google Colab | 3.10 |
| Visual Studio Code Editor | 1.87.2 |
| Python | 3.8 and later |
| Operating System | 64-bit OS, Windows 11 |

### 3.2.1.2 Python packages

Python packages we used in our project includes the following table 2.
We used the command **pip install "*package_name==version*"** to download the packages.

**Table 3: Python packages**

| S.no. | Packages used | Description |
|---|---|---|
| 1. | opencv | It is a popular library used for computer vision tasks, including image and video processing, object detection, and feature extraction. |
| 2. | numpy | Numpy package is fast and accurate, used for scientific computing in python. |
| 3. | pandas | Pandas is a fast, scalable package used for data science operations and expensive data structures operations. |
| 4. | matplotlib | It is a cross-platform, data visualization and graphical plotting library for python and its numerical extension |
| 5. | torch | PyTorch is a deep learning library that provides tools and modules for building and training neural networks. |
| 6. | tempfile | The tempfile module provides functions for creating temporary files and directories. |
| 7. | os | This module provides a portable way of using operating system dependent functionality. |
| 8. | torchvision | The Provides image datasets and models for computer vision, often used with PyTorch. |
| 9. | scikit-learn | A machine learning library for Python, used for classification, regression, and evaluation metrics. |
| 10. | Pillow | Python Imaging Library (PIL) fork used for opening, manipulating, and saving many image file formats. |

| | | |
|---|---|---|
| 11. | csv | The csv module provides classes and functions for reading and writing CSV files. |
| 12. | flask | A micro web framework written in Python, used to build the backend API for cancer prediction. |
| 13. | flask-cors | A Flask extension for handling Cross Origin Resource Sharing (CORS), allowing your backend to connect with a frontend. |
| 14. | io | Python io module allows us to manage the file-related input And output operations. |

### 3.2.2 Hardware Requirements

The below Table 4 shows the minimum hardware requirements of our project.

#### 3.2.2.1 Minimum Hardware Requirements

**Table 4: Hardware Requirements**

| **Hardware Requirements** | **Configuration** |
|---|---|
| Processor | Intel core with i5 configuration or equivalent processors. |
| RAM | 8GB(16 GB recommended for smoother multitasking) |
| Hard Disk | 1 TB HDD or 512 GB SSD |
| GPU (Cloud) | NVIDIA Tesla T4 / P100 / V100 (via Google Colab) |

## 3.3 NON-FUNCTIONAL REQUIREMENTS

- **Compatibility:** According to the definition, Compatibility is the capacity for two systems to work together without having to be altered to do so. This project is compatible to run on Windows 10,11
- **Capacity:** It can be stated as the capacity of a system refers to the amount of storage it utilizes. This project work i5 processor of 8GB RAM
- **Environment:** It can be stated as all the external and internal forces that exert on your project. This project works on python 3.9 environment and higher versions
- **Performance:** The performance of this project is analyzed based on the accuracy of model and confusion matrix.
- **Reliability:** It can be stated as the extent to which the software system consistently performs the specified function without failure. In this project, output is analyzed is based on the dataset which is taken in controlled environment.

## 3.4 FLOWCHART OF OUR PROJECT

Figure 3 illustrates the workflow of the suggested Multi-Cancer Detection System. The system utilizes brain, lung, and breast cancer datasets on Kaggle, which are combined and divided into training, validation, and testing sets. Images are preprocessed through resizing, normalization, flipping, rotation, CLAHE-based color jittering, and transformed to tensors. A ResNet50 model is employed, and the last layer is changed to classify six classes (cancerous and non-cancerous for every type of cancer). The model is trained using Focal Loss, and the weights are stored in a .pth file. For inference, the Flask API loads the model and weights, checks the uploaded image, and returns the cancer classification result through a user interface.

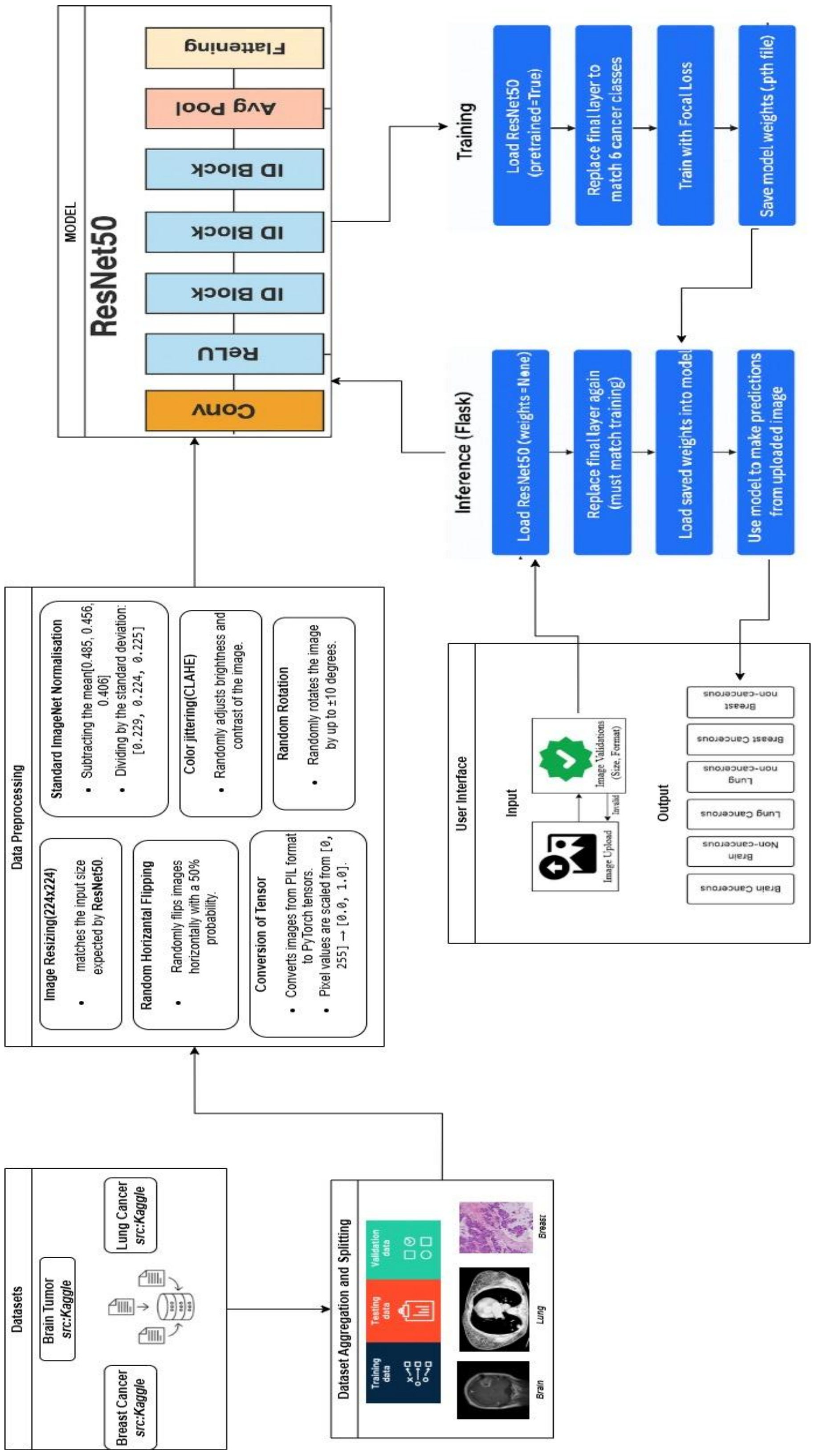

Datasets
Brain Tumor src:Kaggle
Breast Cancer src:Kaggle
Lung Cancer src:Kaggle
Dataset Aggregation and Splitting
Training data
Testing data
Validation data
Brain
Lung
Breast
Data Preprocessing
Image Resizing(224x224)
matches the input size expected by ResNet50.
Random Horizantal Flipping
Randomly flips images horizontally with a 50% probability.
Conversion of Tensor
Converts images from PIL format to PyTorch tensors.
Pixel values are scaled from [0, 255] → [0.0, 1.0].
Standard ImageNet Normalisation
Subtracting the mean[0.485, 0.456, 0.406]
Dividing by the standard deviation: [0.229, 0.224, 0.225]
Color jittering(CLAHE)
Randomly adjusts brightness and contrast of the image.
Random Rotation
Randomly rotates the image by up to ±10 degrees.
MODEL
ResNet50
Conv
ReLU
ID Block
ID Block
ID Block
Avg Pool
Flattening
Training
Load ResNet50 (pretrained=True)
Replace final layer to match 6 cancer classes
Train with Focal Loss
Save model weights (.pth file)
Inference (Flask)
Load ResNet50 (weights=None)
Replace final layer again (must match training)
Load saved weights into model
Use model to make predictions from uploaded image
User Interface
Input
Image Upload
Image Validations (Size, Format)
Invalid
Output
Brain Cancerous
Brain Non-Cancerous
Lung Cancerous
Lung non-cancerous
Breast Cancerous
Breast non-cancerous

**Fig.3: System Architecture**

## 3.5 ALGORITHM

### 3.5.1 ResNet-50

ResNet50 is a convolutional neural network deep architecture extensively used for image classification, most prominently recognized for its capability in coping with deep networks using residual connections. ResNet50 acts as the brain cancer, lung cancer, and breast cancer identifying backbone in the envisaged multi-cancer detection system.

ResNet50 consists of three principal parts:

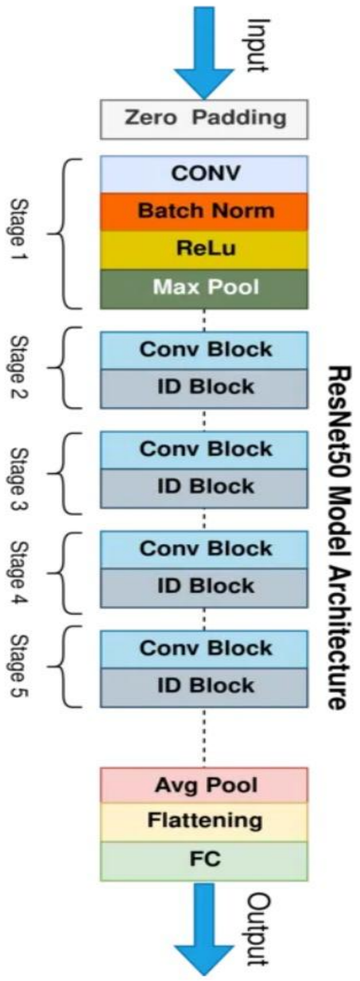


**Fig.4: Architecture of ResNet-50**

#### 3.5.1.1 Backbone

The backbone of ResNet50 performs feature extraction on the input image. It consists of several convolutional layers and residual blocks. The residual blocks incorporate skip connections, through which the model is able to learn identity mappings and hence eliminate the vanishing gradient problem in deep networks. The features extracted contain low-level information like edges and textures and high-level abstract patterns for medical image classification.

#### 3.5.1.2 Neck

The neck is a bridge connecting the backbone to the classification head. It accepts the high-level features learned from the ResNet50 backbone and further processes them to prepare them for the ultimate classification task. This section of the architecture commonly consists of operations like Global Average Pooling, dropout layers, and sometimes other fully connected layers. The Global Average Pooling down-scales the spatial dimensions, summarizing the information into a denser feature vector. Dropout is used to avoid overfitting and enhance generalization. The neck makes sure that the features sent to the head are strong and optimized for cancer type prediction

#### 3.5.1.3 Head

The head of the architecture is the last classification part that provides predictions from the input features. It typically contains one or multiple dense layers, with the last layer having the softmax activation function to produce probabilities over each class of cancer (e.g., brain tumor, breast cancer, lung cancer) or a "no tumor" class. This part maps the feature vector to understandable outputs for the user. The model is trained using focal loss, which is especially efficient in managing class imbalance by placing greater emphasis on hard-to-classify cases. The head therefore makes the final decision as to the presence and the type of cancer in the input image.

### 3.5.2 Working Of ResNet

ResNet50 is a high-performance image classification deep learning model. It is employed here in the case of this multi-cancer detection system to classify medical images into six cancer classes. Here is a step-by-step explanation of how ResNet50 works within the system:

1. **Input Image Processing:**

- The ResNet model takes an input image, the medical image is resized to 224×224 pixels.
- Preprocessing comprises normalization, color enhancement (CLAHE), random flipping, rotation, and tensor format conversion appropriate for PyTorch.

2. **Feature Extraction Backbone:**

- The preprocessed image is fed into a deep CNN that consists of convolutional layers and residual blocks.
- These blocks draw out rich spatial and contextual features required for separating cancerous from non-cancerous tissues.

3. **Residual Connections:**

- Every residual block has a skip connection that adds the block's input to its output, allowing for improved gradient flow and more efficient learning in deeper layers.
- This architecture enables the network to preserve significant information even in extremely deep setups.

4. **Feature Map Generation:**

- The network's output is a dense feature map that represents low-level and high-level visual information.
- A global average pooling layer later transforms this map into a feature vector for the purpose of classification.

5. **Classification Head:**

- The 6-class output layer is customized and makes use of the feature vector to predict the probabilities of every cancer type.
- The most probable class is taken as the prediction output.

6. **Training Process:**

- Training occurs on labeled medical images using Focal Loss, which is biased towards hard-to-classify examples and deals with class imbalance.
- The last trained model is stored as a .pth file for future use during inference.

7. **Inference Prediction:**

- The During the deployment stage, the model weights are loaded into a Flask-based API.
- The user submits a medical image, which is preprocessed and fed into the trained ResNet50 model.
- The model returns the predicted cancer class and a confidence score.

8. **Post Proceessing:**

- The prediction is mapped to the respective cancer category (e.g., Lung Cancerous).
- The output is presented to the user in a web interface in an easily interpretable format.

# 4. DESIGN

## 4.1 INTRODUCTION

Unified modelling language (UML) is used to represent the software in a graphical format that is it provides a standard way to visualize how a system is designed. Good UML design is followed for a better and precise software output. The UML provides different constructs for specifying, visualizing, constructing and documenting the artifacts of software systems. UML diagrams are used to better understand the system, to maintain the document information about the system and emphasizes on the roles, actors, actions, actions and process. The UML diagrams considered are:

- Use-case Diagram
- Activity Diagram
- Sequence Diagram

### 4.1.1 Relationships

The relationships among different entities in the following diagrams are given below and in the figure 12:

- **Association -** This relationship tells how two entities interact with each other.
- **Dependency -** An entity is dependent on another if the change of that is reflecting the other.
- **Aggregation -** It is a special kind of association which exhibits part-of relationship.
- **Generalization-**This relationship describes the parent- child relationship amongdifferent entities.
- **Realization -** It is a kind of relationship in which one thing specifies the behavior or aresponsibility to be carried out, and the other thing carries out that behavior.

## 4.2 UML DIAGRAMS

### 4.2.1 Use-Case Diagram

Use-case diagrams reflect the behavior of a system and assist in highlighting the functional requirements. In the Multi-Cancer Detection System, the diagram depicts how users interact with the system to classify cancer based on medical images, It includes the following components:

4.2.1.1 Actor

4.2.1.2 Use case

#### 4.2.1.1 Actor

An actor is a role that acts on the system. In this project, the main actor is:

User: An individual (e.g., physician, medical technician, or patient) who uploads images to identify potential types of cancer.

#### 4.2.1.2 Use-Case

A use case is a function that a system does to help the user achieve their goal. They are 5 use- cases which has their own functionality that can be useful in accomplishing the goals.

#### 4.2.1.3 Relationships

Solid lines known as associations are used to show the relationships between actors and use cases. A connection signifies an actor's participation in a specific use case. Actors may also exhibit generalization relationships, which indicate specialization or inheritance.

#### 4.2.1.4 System Boundary

IThe system boundary is a box enclosing all use cases. It visually separates the Multi-Cancer Detection System from the external world, indicating which functionalities are handled internally.

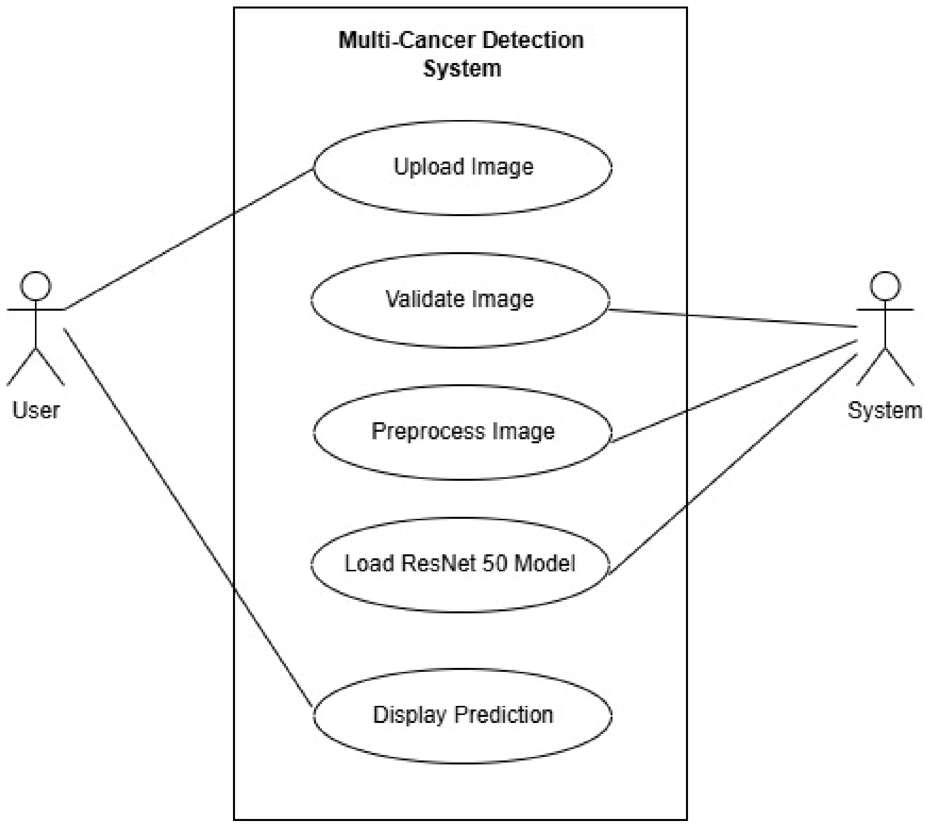


**Fig.5: Use Case Diagram**

### 4.2.2 Sequence Diagram

To be certain, it is important to understand that the sequence diagram (one more name is an event diagram) depicts the course of actions and the information that are transferred between the user and the system when we determine several types of cancer. It gives a chronologically arranged representation of the interactions showing in the process of system functioning (Johnson, 2007).

#### 4.2.2.1 Lifeline

Represents the different participants involved in the sequence. In this diagram, the lifelines are the User and the System (the cancer detection platform)

#### 4.2.2.2 Actor

Indicates the communication between the User and the System when an operation (such as uploading or processing an image) is invoked.

#### 4.2.2.3 Call message

Indicates the communication between the User and the System when an operation (such as uploading or processing an image) is invoked.

#### 4.2.2.4 Return Message

Shows the return of data or results from the System back to the User, such as cancer predictions.

The sequence of messages is shown in the figure 15. The system here represents the GUI that the user interacts with and also predicts that cancer is present or not.

#### 4.2.2.5 Activation

Represented as vertical rectangles, activations indicate when the system is actively processing a message (e.g., analyzing an image).

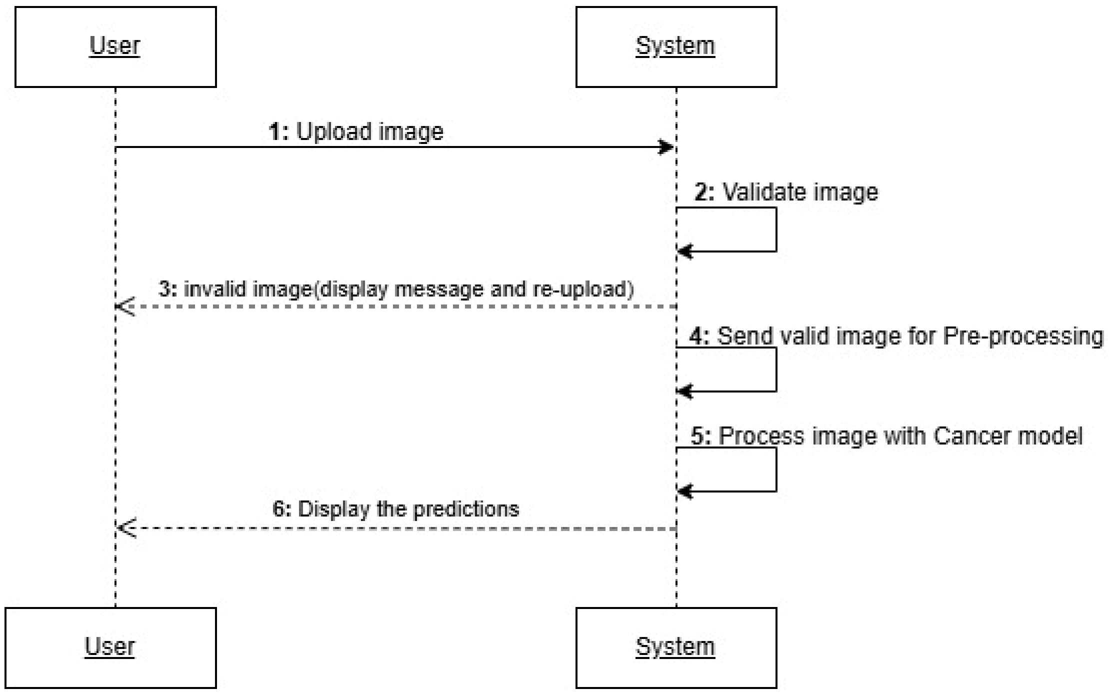

User
System
1: Upload image
2: Validate image
3: invalid image(display message and re-upload)
4: Send valid image for Pre-processing
5: Process image with Cancer model
6: Display the predictions
User
System

**Fig.7: Sequence Diagram**

### 4.2.2 Activity Diagram

In the Unified Modeling Language (UML), an activity diagram is a behavioral diagram that illustrates the flow of control or data through a system. For the multi-cancer detection system, the activity diagram demonstrates the step-by-step process of analyzing a medical image to detect brain, lung, or breast cancer as either cancerous or non-cancerous

The following are an activity diagram's primary elements:

**4.2.3.1 Activities:** Activities in this system represent distinct stages in the image analysis workflow. Examples include uploading the image, validating the scan, preprocessing the image, and passing it through the classification model. These are shown as rounded rectangles.

**4.2.3.2 Actions:** Within an activity, actions are the discrete steps or procedures. They may consist of operations like doing a calculation, sending a message, or calling a procedure. The order in which actions are performed is shown by connecting them with control flow arrows, which are represented as rounded-corner rectangles.

**4.2.3.3 Control Flows:** In a diagram, control flows show how decisions are made between various tasks and actions. They are shown as arrows that connect activities and actions, showing the sequence in which they are carried out. Decision and merge nodes, which indicate branching and merging points in the control flow, can also be included in control flows.

**4.2.3.4 Decision Nodes:** Decision nodes are locations in the diagram where a condition or choice determines how control flows. They appear as diamonds with several outgoing control flows, each of which represents a potential decision-making result.

**4.2.3.5 Merge Nodes:** In a diagram, merge nodes stand for the places where several control flows reunite to form a single flow. They are represented as circles and show pathways in the control flow can merge.

Activity diagrams are useful tools for describing and examining how systems behave, particularly when intricate workflows or business processes are involved.

They offer a visual depiction of the steps and actions that make up a system, assisting stakeholders in comprehending the control flow and spotting any possible inefficiencies or bottlenecks in the system's architecture.

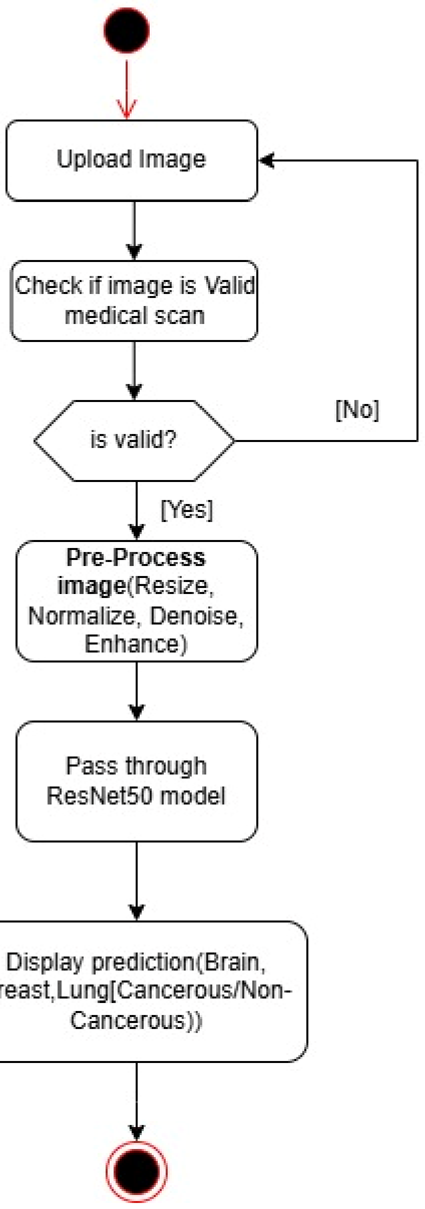

Upload Image
Check if image is Valid medical scan
is valid?
[No]
[Yes]
Pre-Process image(Resize, Normalize, Denoise, Enhance)
Pass through ResNet50 model
Display prediction(Brain, Breast,Lung[Cancerous/Non-Cancerous))

**Fig.8: Activity Diagram**

## 4.3 MODULE DESIGN

### 4.3.1 System Module

The Multi-Cancer Detection System is designed to automate the process of identifying cancerous and non-cancerous conditions from medical MRI scans using a deep learning approach powered by the ResNet-50 model. The system workflow begins when a user uploads a medical image (brain, breast, or lung scan). The image is validated to confirm that it is a proper and analyzable medical scan. If the image is invalid, an error message is displayed, prompting the user to upload a valid scan. Upon validation, the system enters the preprocessing phase, where the image undergoes several enhancement steps such as resizing, normalization, denoising, and contrast adjustment to optimize it for the model. Once prepared, the image is passed through the ResNet-50 model, which performs classification and predicts whether the image represents a cancerous or non-cancerous condition across three categories: brain, lung, and breast. System - Performs validation, preprocessing, classification, and result generation. If invalid, the system immediately displays an error message prompting the user to upload a correct image. Once preprocessing is complete, the image is passed through the ResNet-50 model, which analyzes it and generates a prediction indicating whether a tumor is present. Finally, the system displays the prediction result to the user in a clear and understandable format. The system involves two primary actors: the user, who uploads the image and receives the output, and the system itself, which manages all internal operations from validation to prediction. The modules identified are:

### 4.3.2 Data Collection

MRI, CT and histopathological scan datasets (brain, lung, breast) are collected from reliable medical imaging sources from Kaggle. Aggregated the 3 datasets into one single dataset and split it into train, test and valid folders.

### 4.3.3 Data Pre-processing

Each image is resized to a standard input dimension compatible with ResNet-50. Pixel intensities are normalized to ensure consistency, and techniques such as denoising and contrast enhancement are applied to highlight critical features.

### 4.3.4 Data Annotation

For training purposes, images may be labeled to indicate the type of cancer (if present), including bounding areas of abnormality or labeling the entire scan as cancerous/non-cancerous.

### 4.3.5 Splitting the Data

The dataset is split into training, validation, and testing sets. This ensures proper evaluation and generalization of the model by preventing overfitting and underfitting.

### 4.3.6 Detection Model Building

A fine-tuned ResNet-50 convolutional neural network is used for multi-class classification. It distinguishes between six output classes:

- Brain Cancerous & Non-Cancerous
- Lung Cancerous & Non-Cancerous
- Breast Cancerous & Non-Cancerous

### 4.3.7 Model Evaluation

The model is assessed using performance metrics like accuracy, precision, recall, F1-score, and confusion matrix. These metrics help verify how well the model performs across each category and guide improvements.

## 4.4 CONCLUSION

UML diagrams play a vital role in visualizing the structure and functionality of the Multi-Cancer Detection System. They help in planning and validating the system's flow and logic before implementation.. By outlining the dynamic interactions and static components clearly, UML diagrams for effective cancer prediction.

# 5. IMPLEMENTATION AND RESULTS

## 5.1 INTRODUCTION

The project's implementation stage is when the theoretical design is translated into a workable system. As a result, it can be seen as the most crucial stage in ensuring the success of a new system and giving the user confidence that the system will work and be effective. The implementation step entails meticulous planning, research of the existing system and its implementation limitations, designing of changeover methods, and evaluation of changeover methods.

## 5.2 SOFTWARE DESIGN

### 5.2.1 Front-End Design

#### 5.2.1.1 HTML

Hypertext markup language (HTML) is used for texts intended to be viewed in a web browser. Technologies such as Cascading Style Sheets (CSS) and programming languages like JavaScript can help. Web browsers receive HTML documents from a web server or locally stored files and convert them to multimedia web pages. HTML originally featured cues for the document's look and described the structure of a web page logically. HTML tags are the components that make up HTML pages. Images and other objects, such as interactive forms, can be embedded in the produced page using HTML structures.

HTML allows you to create organized documents by indicating structural semantics for text elements like headers, paragraphs, lists, links, quotations, and other elements. Tags, which are written in angle brackets, separate HTML elements. Tags like image /> and input /> insert content into the page immediately. Other tags, such as p>, surround and offer information about document text and may comprise sub elements of other tags. Browsers do not show HTML tags; instead, they use them to interpret the page's content.

#### 5.2.1.2 CSS

CSS, or Cascading Style Sheets, is a simple design language intended to make the process of making web pages presentable easier. Styles can be applied to web pages using CSS. More crucially, CSS allows you to do so without relying on the HTML that each web page contains. CSS is simple to learn and understand, but it gives you a lot of power over how an HTML document looks. Cascading Style Sheets (CSS) is a key web development tool that defines the appearance and layout of HTML texts. It allows web designers and developers to manage several aspects of a webpage's appearance, including fonts, colors, spacing, and position. CSS operates by choosing HTML components and applying styles to them according to rules stated in style sheets. These style sheets can be incorporated in HTML texts, included externally, or used dynamically with scripting languages. CSS also provides cascading, inheritance, and specificity, which allows for efficient and structured styling across web pages while maintaining design consistency and flexibility.

#### 5.2.1.3 Flask Framework

Flask is an agile and light Python web framework employed in the development of the frontend interface of the Multi-Cancer Detection System. Flask offers a minimal yet robust platform on which web applications can be developed, with easy integration between the machine learning backend and the UI. Flask is utilized to process user interactions like uploading MRI images and showing prediction outcomes. HTML, CSS, and JavaScript are utilized to create the frontend, and Flask is utilized to process routing, request handling, and backend logic. If the user uploads a medical scan using the web interface, Flask receives the request, sends the image to the ResNet-50 model to be analyzed, and sends the prediction (for instance, brain, lung, or breast cancer — cancerous or non-cancerous) back. The result is then displayed on the webpage for easy visualization. This configuration supports successful deployment of the machine learning model to a real-world web environment so that it can be accessed by users through a browser-based interface.

### 5.2.2 Backend-Design

#### 5.2.2.1 Data Collection

The datasets which are used in this project is extracted from Kaggle datasets. We used three datasets(Brain MRI scans dataset, Lung CT scans, for training one model. The following table explains the structure of the dataset.

**Table 5: Information about Datasets**

| **Dataset** | **Total Images** | **Training Set** | **Validation Set** |
|---|---|---|---|
| Brain Tumor MRI Scans Dataset | 5,585 | 3,919 | 573 |
| Lung Cancer CT Scans Dataset | 1,000 | 690 | 101 |
| Breast Histopathological Scans Dataset | 1,213 | 848 | 122 |

#### 5.2.2.2 Data Preprocessing

Preprocessing techniques such as resizing, normalization, and augmentation play a critical role in computer vision-based cancer detection. In this system, all input MRI, CT, and X-ray images are resized to a uniform resolution to maintain consistency across the dataset. Normalization is applied to scale the pixel values, typically between 0 and 1, which aids in faster convergence and improved model performance. Augmentation techniques like rotation, flipping, and zooming are applied to artificially increase the dataset size, improve model generalization, and reduce overfitting. These steps ensure that the input to the ResNet-50 model is clean, consistent, and informative for accurate cancer prediction.

#### 5.2.2.3 Data Annotation

Data annotation is carried out by assigning class labels to each image in the dataset. The labels correspond to six distinct classes: brain cancerous, brain non-cancerous, lung cancerous, lung non-cancerous, breast cancerous, and breast non-cancerous. This labeling is typically done by referencing clinical datasets with verified ground truth or annotations provided by medical experts. These class labels are encoded and used to train the deep learning model in a supervised learning setup.

#### 5.2.2.4 Model Building

A customized ResNet-50 model, fine-tuned from its ImageNet pre-trained version, is used for multi-class cancer classification. The final layer is modified to output six classes. The model is trained using Focal Loss to address class imbalance, and optimized with Adam in PyTorch, incorporating suitable learning rates and regularization to prevent overfitting. The trained ResNet-50 model is deployed using the Flask framework, providing a simple web interface where users (e.g., doctors) can upload medical images. Once uploaded, the backend preprocesses the image and passes it to the model, which returns the predicted cancer class with a confidence score. The result is then displayed to the user through the interface.

## 5.2 IMPLEMENTATION

### 5.3.1 Front-End

**index.html**

```
<!DOCTYPE html>
<html lang="en" data-theme="light">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Multi-Cancer Detection System</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&di
splay=swap" rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/aos/2.3.4/aos.css">
</head>
<body>
  <nav class="navbar">
    <div class="nav-brand">
      <a href="/">Multi-Cancer Detection System</a>
    </div>
    <div class="nav-links">
      <a href="/">Home</a>
      <a href="/detection">Detection</a>
      <a href="/dashboard">Dashboard</a>
      <a href="/about">About</a>
      <a href="/cancer-types">Cancer Types</a>
      <button id="themeToggle" class="theme-toggle" title="Toggle dark mode">
        <i class="fas fa-moon"></i>
      </button>
    </div>
  </nav>

  <div class="container">
    <!-- Hero Section -->
    <section class="hero-section">
      <div class="hero-content" data-aos="fade-up">
        <h1>Early Detection Saves Lives</h1>
        <p class="hero-subtitle">Get instant cancer detection using our AI-powered
system</p>
        <div class="hero-buttons">
          <a href="/detection" class="primary-button">Start Detection</a>
```

```
                <a href="/about" class="secondary-button">Learn More</a>
            </div>
        </div>
        <div class="hero-image" data-aos="fade-left">
            <img src="{{ url_for('static', filename='images/hero-image.jpg') }}" alt="Cancer
Detection Illustration">
        </div>
    </section>

    <!-- Statistics Section -->
    <section class="stats-section" data-aos="fade-up">
        <div class="stats-container">
            <div class="stat-card">
                <div class="stat-icon"><i class="fas fa-chart-line"></i></div>
                <div class="stat-number" data-count="95">0</div>
                <div class="stat-label">Accuracy Rate</div>
            </div>
            <div class="stat-card">
                <div class="stat-icon"><i class="fas fa-brain"></i></div>
                <div class="stat-number" data-count="3">0</div>
                <div class="stat-label">Cancer Types</div>
            </div>
            <div class="stat-card">
                <div class="stat-icon"><i class="fas fa-clock"></i></div>
                <div class="stat-number" data-count="10">0</div>
                <div class="stat-label">Second Analysis</div>
            </div>
        </div>
    </section>

    <!-- Features Section -->
    <section class="features-section">
        <h2 class="section-title" data-aos="fade-up">Why Choose Our System</h2>
        <div class="features-grid">
            <div class="feature-card" data-aos="fade-up" data-aos-delay="100">
                <div class="feature-icon">
                <i class="fas fa-microscope"></i>
                </div>
                <h3>Advanced Detection</h3>
                <p>Using state-of-the-art AI models trained on thousands of medical scans for
accurate cancer detection</p>
            </div>
            <div class="feature-card" data-aos="fade-up" data-aos-delay="200">
                <div class="feature-icon">
                <i class="fas fa-clock"></i>
                </div>
                <h3>Quick Results</h3>
                <p>Get instant results within seconds, allowing for faster medical intervention
when needed</p>
            </div>
            <div class="feature-card" data-aos="fade-up" data-aos-delay="300">
                <div class="feature-icon">
                <i class="fas fa-shield-alt"></i>
                </div>
                <h3>Reliable</h3>
                <p>High accuracy in detecting various types of cancer with detailed
confidence scores</p>
            </div>
            <div class="feature-card" data-aos="fade-up" data-aos-delay="400">
                <div class="feature-icon">
                    <i class="fas fa-chart-pie"></i>
                </div>
                <h3>Comprehensive Analysis</h3>
                <p>Detailed insights and visualizations to understand the detection results</p>
            </div>
            <div class="feature-card" data-aos="fade-up" data-aos-delay="500">
                <div class="feature-icon">
                    <i class="fas fa-lock"></i>
                </div>
                <h3>Privacy Focused</h3>
                <p>Your medical data is processed securely and never stored without
permission</p>
            </div>
            <div class="feature-card" data-aos="fade-up" data-aos-delay="600">
                <div class="feature-icon">
                    <i class="fas fa-mobile-alt"></i>
                </div>
                <h3>Mobile Friendly</h3>
                <p>Access our system from any device, anywhere, at any time</p>
            </div>
        </div>
    </section>

    <!-- How It Works Section -->
    <section class="how-it-works-section">
        <h2 class="section-title" data-aos="fade-up">How It Works</h2>
        <div class="steps-container">
            <div class="step" data-aos="fade-right" data-aos-delay="100">
```

```
                <div class="step-number">1</div>
                <div class="step-content">
                    <h3>Upload Your Scan</h3>
                    <p>Upload your medical scan image <p>
                </div>
            </div>
            <div class="step" data-aos="fade-right" data-aos-delay="200">
                <div class="step-number">2</div>
                <div class="step-content">
                    <h3>AI Analysis</h3>
                    <p>Our advanced AI model analyzes the image for signs of cancer</p>
                </div>
            </div>
            <div class="step" data-aos="fade-right" data-aos-delay="300">
                <div class="step-number">3</div>
                <div class="step-content">
                    <h3>Get Results</h3>
                    <p>Receive instant results with confidence scores and detailed analysis</p>
                </div>
            </div>
            <div class="step" data-aos="fade-right" data-aos-delay="400">
                <div class="step-number">4</div>
                <div class="step-content">
                    <h3>Consult Healthcare Provider</h3>
                    <p>Share results with your healthcare provider for professional medical
advice</p>
                </div>
            </div>
        </div>
    </section>
    <!-- CTA Section -->
    <section class="cta-section" data-aos="fade-up">
        <div class="cta-content">
            <h2>Ready to Get Started?</h2>
            <p>Upload your medical scan for instant cancer detection</p>
            <a href="/detection" class="cta-button">Start Detection Now</a>
        </div>
        <div class="cta-image">
            <img src="{{ url_for('static', filename='images/cta-image.jpg') }}" alt="Start
Detection">
        </div>
    </section>
    <footer>
        <div class="footer-content">
```

```
            <div class="footer-section">
                <h4>Quick Links</h4>
                <ul>
                    <li><a href="/">Home</a></li>
                    <li><a href="/detection">Detection</a></li>
                    <li><a href="/dashboard">Dashboard</a></li>
                    <li><a href="/about">About</a></li>
                    <li><a href="/cancer-types">Cancer Types</a></li>
                    <li><a href="/contact">Contact</a></li>
                </ul>
            </div>
            <div class="footer-section">
                <h4>Contact Us</h4>
                <p><i class="fas fa-envelope"></i> support@cancerdetection.com</p>
                <p><i class="fas fa-phone"></i> +1 234 567 890</p>
                <p><i class="fas fa-map-marker-alt"></i> 123 Medical Center Dr, Healthcare
City</p>
            </div>
            <div class="footer-section">
                <h4>Follow Us</h4>
                <div class="social-links">
                    <a href="#"><i class="fab fa-facebook"></i></a>
                    <a href="#"><i class="fab fa-twitter"></i></a>
                    <a href="#"><i class="fab fa-linkedin"></i></a>
                    <a href="#"><i class="fab fa-instagram"></i></a>
                </div>
            </div>
        </div>
        <div class="footer-bottom">
            <p>Disclaimer: This tool is for educational purposes only. Always consult with
healthcare professionals for medical advice.</p>
            <p>&copy; 2023 Cancer Detection System. All rights reserved.</p>
        </div>
    </footer>
  </div>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/aos/2.3.4/aos.js"></script>
  <script src="{{ url_for('static', filename='js/main.js') }}"></script>
  <script src="{{ url_for('static', filename='js/theme-test.js') }}"></script>
  <script src="{{ url_for('static', filename='js/home.js') }}"></script>
</body>
</html>
```

**5.3.2 Multi-Cancer Detection Model**

**Model.ipynb**

```
from google.colab import drive
drive.mount('/content/drive')
import torch
import torch.nn as nn
import torchvision.models as models
import torchvision.transforms as transforms
from torch.utils.data import Dataset, DataLoader, WeightedRandomSampler
from PIL import Image
import os
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import numpy as np
from collections import Counter
import torch.nn.functional as F
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
class FocalLoss(nn.Module):
    def __init__(self, alpha=1, gamma=2):
        super(FocalLoss, self).__init__()
        self.alpha = alpha
        self.gamma = gamma
    def forward(self, inputs, targets):
        ce_loss = F.cross_entropy(inputs, targets, reduction='none')
        pt = torch.exp(-ce_loss)
        focal_loss = self.alpha * (1-pt)**self.gamma * ce_loss
        return focal_loss.mean()
class CancerDataset(Dataset):
    def __init__(self, root_dir, transform=None, mode='train'):
        self.root_dir = root_dir
        self.transform = transform
        self.mode = mode
        self.classes = ['brain_cancerous', 'brain_non_cancerous',
                      'lung_cancerous', 'lung_non_cancerous',
                      'breast_cancerous', 'breast_non_cancerous']
        self.class_to_idx = {cls: idx for idx, cls in enumerate(self.classes)}
        self.images = []
        self.labels = []
        print(f"Loading {mode} dataset...")
        for class_name in self.classes:
            class_path = os.path.join(root_dir, class_name)
            if os.path.exists(class_path):
```

```
            image_files = [f for f in os.listdir(class_path)
                       if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
            if mode == 'train':
                image_files = image_files[:int(0.8*len(image_files))]
            else:
                image_files = image_files[int(0.8*len(image_files)):]
            self.images.extend([os.path.join(class_path, img) for img in image_files])
            self.labels.extend([self.class_to_idx[class_name]] * len(image_files))
        label_counts = Counter(self.labels)
        total_samples = len(self.labels)
        self.weights = [total_samples/(len(self.classes) * label_counts[label])
                    for label in self.labels]
        print(f"Loaded {len(self.images)} images for {mode}")
        print("Class distribution:", dict(label_counts))
    def __len__(self):
        return len(self.images)
    def __getitem__(self, idx):
        img_path = self.images[idx]
        try:
            image = Image.open(img_path).convert('RGB')
            if self.transform:
                image = self.transform(image)
            return image, self.labels[idx]
        except Exception as e:
            print(f"Error loading image {img_path}: {str(e)}")
            return torch.zeros((3, 224, 224)), self.labels[idx]
class ResNet50Model(nn.Module):
    def __init__(self, num_classes=6):
        super(ResNet50Model, self).__init__()
        self.resnet = models.resnet50(pretrained=True)
        num_features = self.resnet.fc.in_features
        # Freeze early layers
        for param in list(self.resnet.parameters())[:-6]:
            param.requires_grad = False
        self.resnet.fc = nn.Sequential(
            nn.Linear(num_features, 512),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(512, num_classes))

    def forward(self, x):
        return self.resnet(x)
def preprocess_and_validate_image(image_path):
    try:
        img = cv2.imread(image_path)
        if img is None:
            return None, "Invalid image file"
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        min_dimension = 224
        height, width = img.shape[:2]
        if height < min_dimension or width < min_dimension:
            scale_factor = min_dimension / min(height, width)
            new_height = int(height * scale_factor)
            new_width = int(width * scale_factor)
            img = cv2.resize(img, (new_width, new_height),
interpolation=cv2.INTER_CUBIC)
        # Enhance image quality
        img = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)
        lab = cv2.cvtColor(img, cv2.COLOR_RGB2LAB)
        l, a, b = cv2.split(lab)
        clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
        l = clahe.apply(l)
        lab = cv2.merge((l,a,b))
        img = cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        hist = cv2.calcHist([gray], [0], None, [256], [0, 256])
        hist_norm = hist.ravel() / hist.sum()
        mean_intensity = np.mean(gray)
        std_intensity = np.std(gray)
        if std_intensity < 10:
            return None, "Image appears to be too uniform, not a medical scan"
        if mean_intensity < 20 or mean_intensity > 235:
            img = cv2.convertScaleAbs(img, alpha=1.5, beta=10)
            gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
            mean_intensity = np.mean(gray)
            if mean_intensity < 20 or mean_intensity > 235:
                return None, "Image too bright or too dark"
        return img, "Valid medical image"
    except Exception as e:
        return None, f"Error processing image: {str(e)}"
def train_model(model, train_loader, val_loader, criterion, optimizer, device,
num_epochs=5):
    best_val_acc = 0
    history = {
        'train_loss': [], 'train_acc': [],
        'val_loss': [], 'val_acc': []
    }
    for epoch in range(num_epochs):
```

```
print(f'\nEpoch {epoch+1}/{num_epochs}')
print('-' * 10)
# Training phase
model.train()
train_loss = 0
train_correct = 0
train_total = 0
pbar = tqdm(train_loader, desc='Training')
for inputs, labels in pbar:
    inputs, labels = inputs.to(device), labels.to(device)
    optimizer.zero_grad()
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
    train_loss += loss.item()
    _, predicted = outputs.max(1)
    train_total += labels.size(0)
    train_correct += predicted.eq(labels).sum().item()
    pbar.set_postfix({'loss': train_loss/train_total,
                    'acc': 100.*train_correct/train_total})
train_loss = train_loss/len(train_loader)
train_acc = 100.*train_correct/train_total
# Validation phase
model.eval()
val_loss = 0
val_correct = 0
val_total = 0

with torch.no_grad():
    for inputs, labels in tqdm(val_loader, desc='Validation'):
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        val_loss += loss.item()
        _, predicted = outputs.max(1)
        val_total += labels.size(0)
        val_correct += predicted.eq(labels).sum().item()

val_loss = val_loss/len(val_loader)
val_acc = 100.*val_correct/val_total
print(f'Train Loss: {train_loss:.4f} Acc: {train_acc:.2f}%')
print(f'Val Loss: {val_loss:.4f} Acc: {val_acc:.2f}%')
# Save best model
```

```
            if val_acc > best_val_acc:
                best_val_acc = val_acc
                torch.save({
                    'epoch': epoch,
                    'model_state_dict': model.state_dict(),
                    'optimizer_state_dict': optimizer.state_dict(),
                    'val_acc': val_acc,_+
                }, 'best_cancer_model.pth')
            # Update history
            history['train_loss'].append(train_loss)
            history['train_acc'].append(train_acc)
            history['val_loss'].append(val_loss)
            history['val_acc'].append(val_acc)
        return history
    def plot_training_history(history):
        plt.figure(figsize=(12, 4))
        plt.subplot(1, 2, 1)
        plt.plot(history['train_loss'], label='Train Loss')
        plt.plot(history['val_loss'], label='Val Loss')
        plt.title('Model Loss')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
        plt.legend()
        plt.subplot(1, 2, 2)
        plt.plot(history['train_acc'], label='Train Acc')
        plt.plot(history['val_acc'], label='Val Acc')
        plt.title('Model Accuracy')
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy (%)')
        plt.legend()
        plt.tight_layout()
        plt.savefig('training_history.png')
        plt.close()

    class CancerPredictor:
        def __init__(self, model_path='best_cancer_model.pth'):
            self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
            self.model = ResNet50Model().to(self.device)
            # Load the saved model
            checkpoint = torch.load(model_path, map_location=self.device)
            self.model.load_state_dict(checkpoint['model_state_dict'])
            self.model.eval()
            self.classes = ['Brain Cancer', 'Brain Normal',
                       'Lung Cancer', 'Lung Normal',
                       'Breast Cancer', 'Breast Normal']

            self.transform = transforms.Compose([
                transforms.Resize((224, 224)),
                transforms.ToTensor(),
                transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                  std=[0.229, 0.224, 0.225])
            ])
        def predict(self, image_path):
            try:
                processed_image, message = preprocess_and_validate_image(image_path)
                if processed_image is None:
                    return {"error": message}
                image = Image.fromarray(processed_image)
                image = self.transform(image).unsqueeze(0).to(self.device)
                with torch.no_grad():
                    outputs = self.model(image)
                    probabilities = F.softmax(outputs, dim=1)
                    predicted_class = torch.argmax(probabilities, dim=1).item()
                    confidence = float(probabilities[0][predicted_class])
                return {
                    "class": self.classes[predicted_class],
                    "confidence": f"{confidence:.2%}",
                    "status": "success",
                    "preprocessing_message": message
                }
            except Exception as e:
                return {"error": f"Error processing image: {str(e)}"}

    def test_preprocessing(image_path):
        original = cv2.imread(image_path)
        original = cv2.cvtColor(original, cv2.COLOR_BGR2RGB)
        processed, message = preprocess_and_validate_image(image_path)
        if processed is not None:
            plt.figure(figsize=(12, 4))
            plt.subplot(121)
            plt.imshow(original)
            plt.title('Original Image')
            plt.axis('off')
            plt.subplot(122)
            plt.imshow(processed)
            plt.title('Preprocessed Image')
            plt.axis('off')
            plt.savefig('preprocessing_result.png')
            plt.close()
            print(f"Preprocessing message: {message}")
        else:
            print(f"Preprocessing failed: {message}")
    def main():
        # Set device
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        print(f"Using device: {device}")
        # Data transforms
        train_transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.RandomHorizontalFlip(),
            transforms.RandomRotation(10),
            transforms.ColorJitter(brightness=0.2, contrast=0.2),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406],
                        std=[0.229, 0.224, 0.225])])
        val_transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406],
                        std=[0.229, 0.224, 0.225])
        ])
        # Create datasets
        train_dataset =
    CancerDataset(root_dir='/content/drive/MyDrive/theogdataset/train',
    transform=train_transform, mode='train')
        val_dataset = CancerDataset(root_dir='/content/drive/MyDrive/theogdataset/valid',
    transform=val_transform, mode='val')
        # Create weighted sampler
        sampler = WeightedRandomSampler(
            weights=train_dataset. weights,
            num_samples=len(train_dataset),
            replacement=True)
        # Create data loaders
        train_loader = DataLoader(
            train_dataset, batch_size=32, num_workers=4, pin_memory=True)
        val_loader = DataLoader(
            val_dataset,
            batch_size=32,
            shuffle=False,
            num_workers=4,
            pin_memory=True)
        # Initialize model
        model = ResNet50Model().to(device)
        # Initialize focal loss and optimizer
        criterion = FocalLoss()
        optimizer = torch.optim.AdamW(model.parameters(), lr=0.001)
        # Train model
        print("Starting training...")
        history = train_model(model, train_loader, val_loader, criterion, optimizer, device)
        # Plot training history
        plot_training_history(history)
        print("Training completed! Model saved as 'best_cancer_model.pth'")
    if __name__ == "__main__":
        # For training the model
        main()
```

### 5.3.3 Integration Model

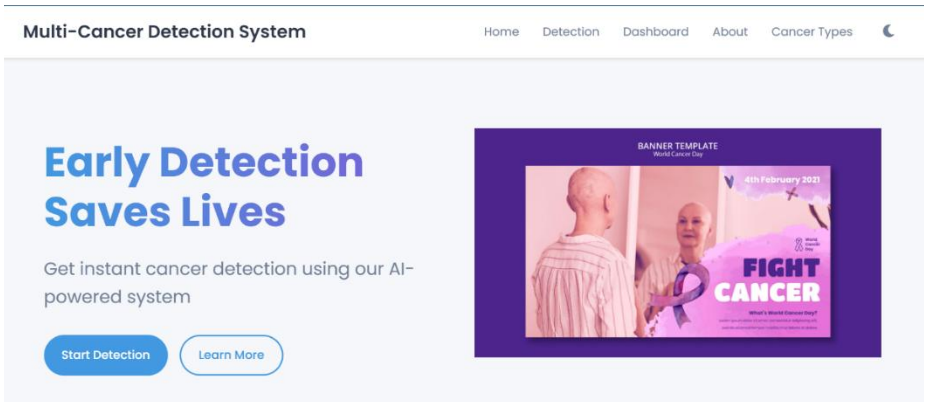
**App.py**

```
from flask import Flask, render_template, request, jsonify
import torch  # Changed from TensorFlow to PyTorch
import numpy as np
from PIL import Image
import io
import os
from torchvision import transforms
import torch.nn.functional as F
import cv2
app = Flask(__name__)
class ResNet50Model(torch.nn.Module):
    def __init__(self, num_classes=6):
        super().__init__()
        self.resnet = torch.hub.load('pytorch/vision:v0.10.0', 'resnet50', weights=None)
        num_features = self.resnet.fc.in_features
        self.resnet.fc = torch.nn.Sequential(
            torch.nn.Linear(num_features, 512),
            torch.nn.ReLU(),
            torch.nn.Dropout(0.5),
            torch.nn.Linear(512, num_classes))
    def forward(self, x):
        return self.resnet(x)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = ResNet50Model().to(device)
model.load_state_dict(torch.load('trailbest_cancer_model.pth',
map_location=device)["model_state_dict"])
```

```
model.eval()
# Define cancer types (6 classes)
CANCER_TYPES = [
    'Brain Cancerous', 'Brain Non-Cancerous',
    'Lung Cancerous', 'Lung Non-Cancerous',
    'Breast Cancerous', 'Breast Non-Cancerous']
# Preprocessing transform
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
def is_medical_scan(image):
    # Convert PIL Image to numpy array
    img_array = np.array(image)
    if len(img_array.shape) == 3:
        # Convert to HSV color space which better separates color information
        if img_array.shape[2] == 3:  # Ensure it's an RGB image
            img_hsv = cv2.cvtColor(img_array, cv2.COLOR_RGB2HSV)
            h_hist = cv2.calcHist([img_hsv], [0], None, [30], [0, 180])
            s_hist = cv2.calcHist([img_hsv], [1], None, [32], [0, 256])
            # Check for typical H&E staining patterns in histopathology
            # Histopathology images often have peaks in specific hue ranges (purples/pinks)
            h_hist_normalized = h_hist / np.sum(h_hist)
            s_hist_normalized = s_hist / np.sum(s_hist)
            # Calculate the average saturation
            avg_saturation = np.mean(img_hsv[:,:,1])
            # Check for H&E staining patterns (purple/pink hues)
            purple_pink_range = np.sum(h_hist_normalized[20:30])  # Hues in purple/pink
range
            if purple_pink_range > 0.3:  # If significant purple/pink colors present
                return True
            # Histopathology images typically have medium to high saturation
            if avg_saturation > 40:  # Lowered threshold from 50 to 40
                return True
            # Check for grayscale characteristics (for CT/MRI)
            r, g, b = img_array[:,:,0], img_array[:,:,1], img_array[:,:,2]
            rg_diff = np.mean(np.abs(r - g))
            rb_diff = np.mean(np.abs(r - b))
            gb_diff = np.mean(np.abs(g - b))
            avg_diff = (rg_diff + rb_diff + gb_diff) / 3
            if avg_diff < 35:  # Increased threshold from 30 to 35
                return True
            # Additional check for uniform background with distinct foreground (common in
medical images)
            gray = cv2.cvtColor(img_array, cv2.COLOR_RGB2GRAY)
            _, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
            white_percentage = np.sum(binary == 255) / binary.size
            # Many medical images have significant white/black background
            if white_percentage > 0.65 or white_percentage < 0.35:  # Adjusted thresholds
                return True
            return False
    return True  # Already grayscale
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/detection')
def detection():
    return render_template('detection.html')
@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/cancer-types')
def cancer_types():
    return render_template('cancer_types.html')
@app.route('/dashboard')
def dashboard():
    return render_template('dashboard.html')
@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get the image from POST request
        if 'file' not in request.files:
            return jsonify({'status': 'error', 'message': 'No file uploaded'}), 400
        file = request.files['file']
        if file.filename == '':
            return jsonify({'status': 'error', 'message': 'No selected file'}), 400
        # Open and validate image
        image = Image.open(io.BytesIO(file.read()))
        if image.mode != 'RGB':
            image = image.convert('RGB')
        # Check if the image appears to be a medical scan
        if not is_medical_scan(image):
            return jsonify({
                'status': 'error',
                'message': 'Please upload a medical scan image. The image appears to be a
regular photo or artwork.'
            }), 400
        # Preprocess the image
        image_tensor = transform(image).unsqueeze(0).to(device)
        # Make prediction
        with torch.no_grad():
            outputs = model(image_tensor)
            probabilities = F.softmax(outputs, dim=1)
            confidence, predicted_class = torch.max(probabilities, 1)
            confidence = confidence.item()
            predicted_class = predicted_class.item()
        # Add confidence threshold check
        if confidence < 0.80:  # You can adjust this threshold as needed
            return jsonify({
                'status': 'error',
                'message': 'Unable to confidently classify this image. Please ensure you are
uploading a clear medical scan.'
            }), 400
        # Prepare response
        response = {
            'cancer_type': CANCER_TYPES[predicted_class],
            'confidence': f"{confidence:.2%}",
            'status': 'success'}
    except Exception as e:
        print("Error during prediction:", str(e))
        response = {
            'status': 'error',
            'message': 'Error processing image. Please upload a valid brain, breast, or lung
scan.'
        }
        return jsonify(response), 500
    return jsonify(response)
if __name__ == '__main__':
    app.run(debug=True)
```

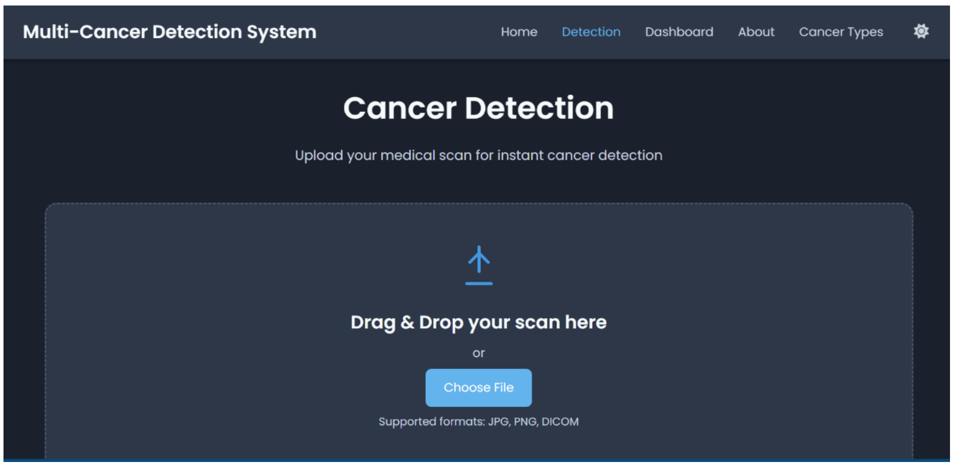
## 5.3 OUTPUT SCREENS

After successful execution of this project, the early detection of brain, lung, and breast cancers from medical images was achieved using AI-based analysis. The system provides accurate classification of cancerous and non-cancerous cases, helping in timely diagnosis and treatment. The Home page, shown in the image, allows users to navigate to other sections such as Detection, Dashboard, About, and Cancer Types, making the system easily accessible in Screen 1.


Multi-Cancer Detection System
Home
Detection
Dashboard
About
Cancer Types
Early Detection Saves Lives
Get instant cancer detection using our AI-powered system
Start Detection
Learn More
BANNER TEMPLATE
4th February 2021
FIGHT CANCER

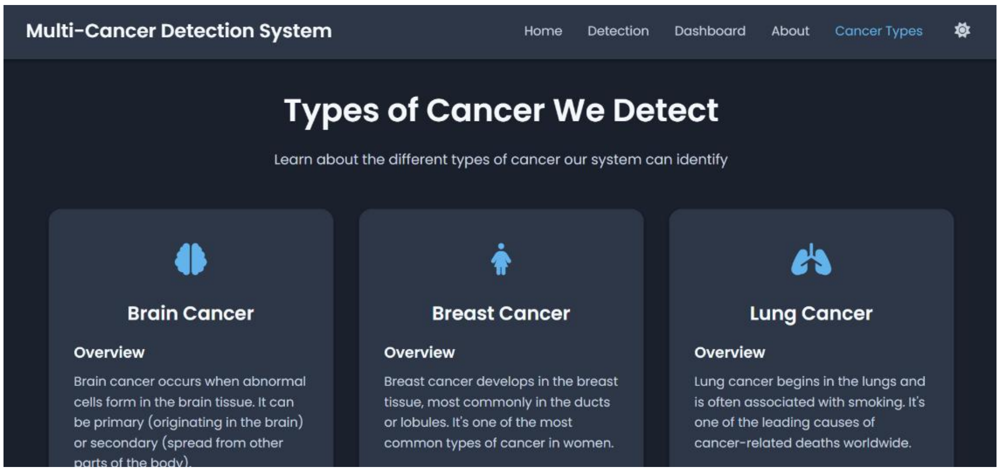
**Screen 1 - Home Page**

The Detection page allows users to upload medical scans in JPG, PNG, or DICOM format for instant AI-powered cancer analysis in Screen 2.


Multi-Cancer Detection System
Home
Detection
Dashboard
About
Cancer Types
Cancer Detection
Upload your medical scan for instant cancer detection
Drag & Drop your scan here
or
Choose File
Supported formats: JPG, PNG, DICOM

**Screen 2 - Detection Page**

The Cancer Types page provides an overview of the different cancers—brain, breast, and lung—that our system can detect and classify in Screen 3.



Types of Cancer We Detect

Learn about the different types of cancer our system can identify

Brain Cancer

Overview

Brain cancer occurs when abnormal cells form in the brain tissue. It can be primary (originating in the brain) or secondary (spread from other parts of the body).

Breast Cancer

Overview

Breast cancer develops in the breast tissue, most commonly in the ducts or lobules. It's one of the most common types of cancer in women.

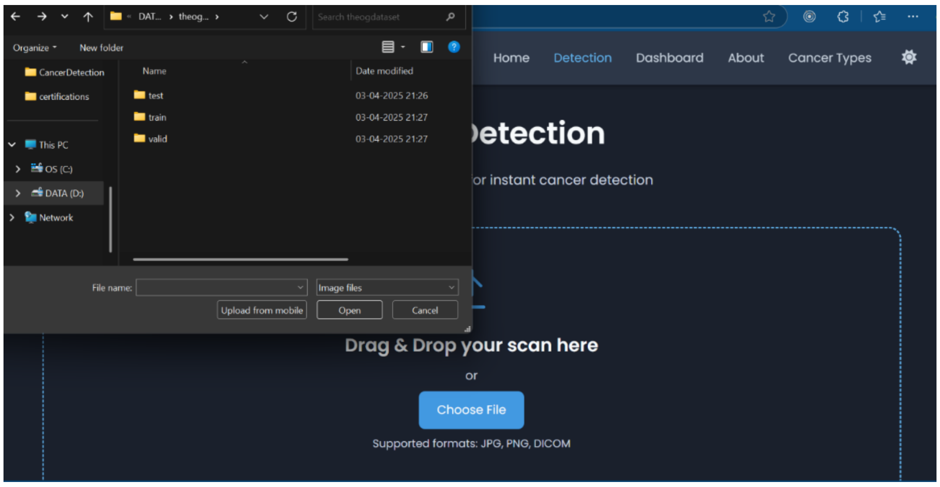
Lung Cancer

Overview

Lung cancer begins in the lungs and is often associated with smoking. It's one of the leading causes of cancer-related deaths worldwide.

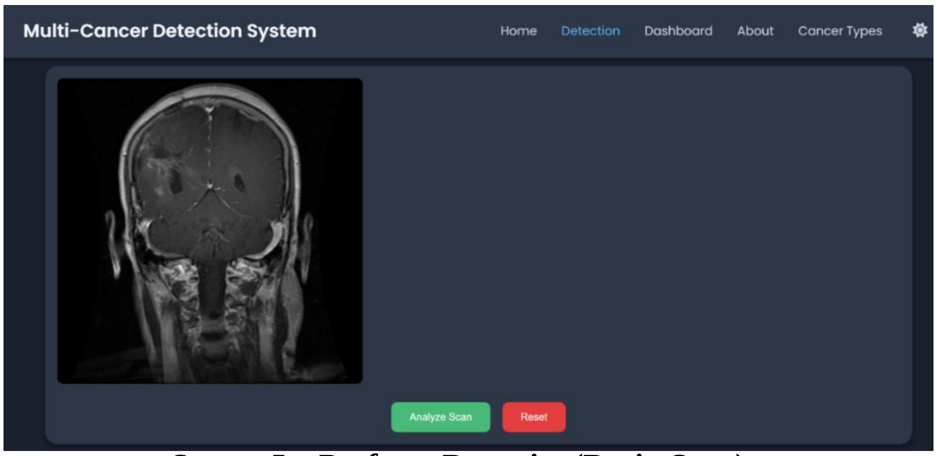
**Screen 3 – Cancer Types**

User is in the process of selecting an image file from their local directory (with folders named test, train, and valid) to upload for analysis. The interface includes a "Choose File" button and supports image formats like JPG, PNG, and DICOM in screen 4.



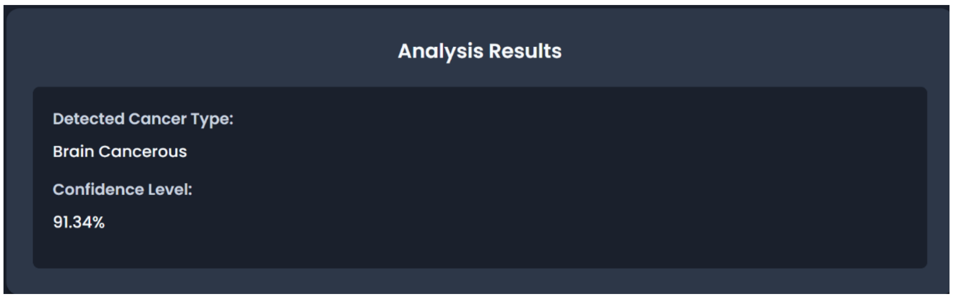
**Screen 4 - Upload Medical Scan**

A brain MRI scan has been uploaded and displayed. Below the scan, there are two buttons labeled "Analyze Scan" and "Reset," allowing the user to initiate cancer detection or clear the current input in Screen 5.



**Screen 5 – Perform Detection(Brain Scan)**

The uploaded scan was classified as "Brain Cancerous" with a confidence level of 91.34%. This suggests a high likelihood of cancer presence in the analyzed brain scan in Screen 6.



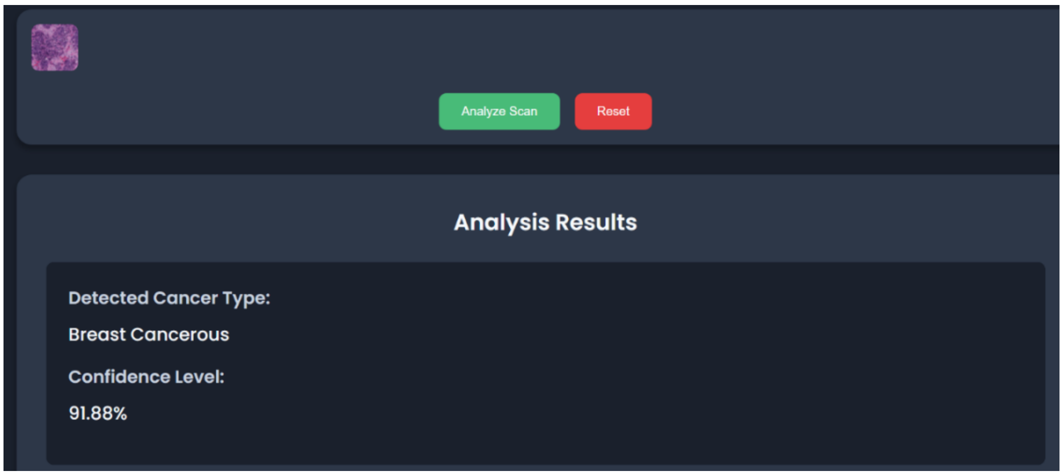
Analysis Results

Detected Cancer Type:
Brain Cancerous

Confidence Level:
91.34%

**Screen 6 - Detection on the WebPage**

A Breast scan has been uploaded and displayed. Below the scan, there are two buttons labeled "Analyze Scan" and "Reset," allowing the user to initiate cancer detection or clear the current input. The uploaded scan was classified as "Breast Cancerous" with a confidence level of 91.8%. This suggests a high likelihood of cancer presence in the analyzed breast scan in Screen 7.



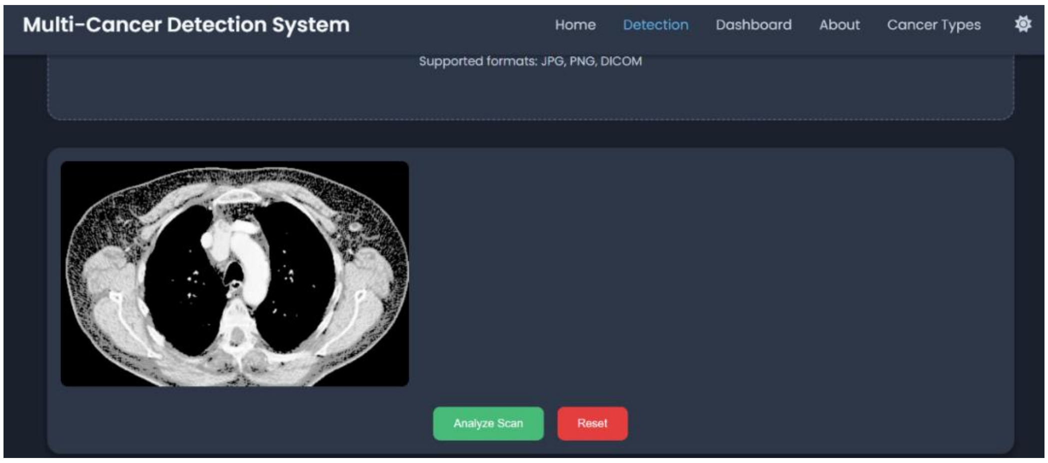
Analysis Results

Detected Cancer Type:
Breast Cancerous

Confidence Level:
91.88%

**Screen 7 - Perform Detection(Breast Cancer)**

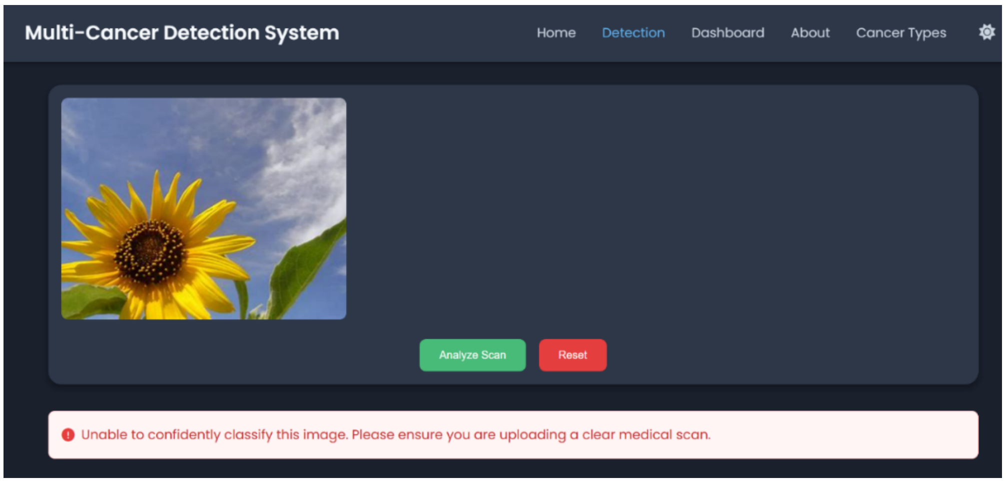
A Lung CT scan has been uploaded and displayed. Below the scan, there are two buttons labeled "Analyze Scan" and "Reset," allowing the user to initiate cancer detection or clear the current input in Screen 8.



Supported formats: JPG, PNG, DICOM

**Screen 8 - Perform Detection(Lung Cancer)**

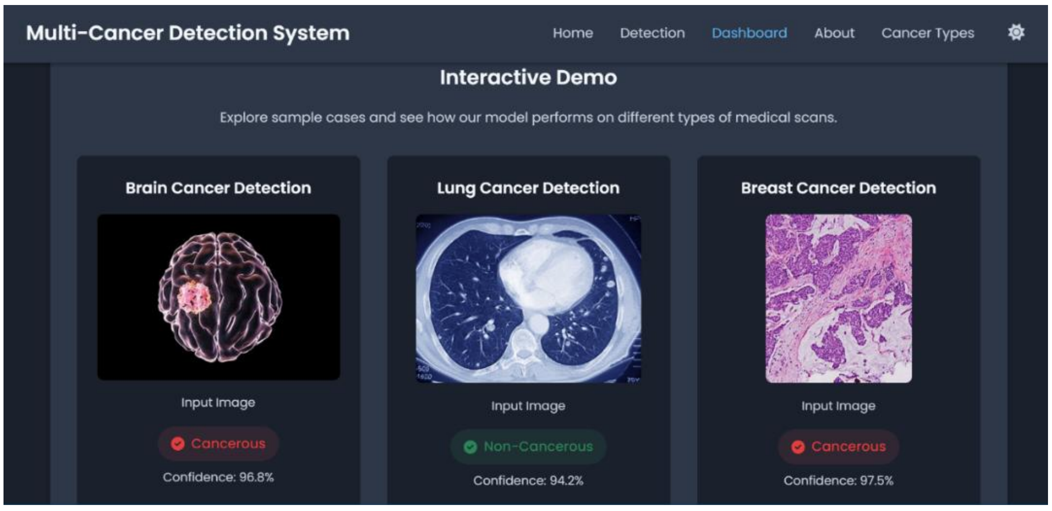
The invalid input scenario, where a non-medical image (a sunflower) was uploaded for cancer detection. The system responds with an error message, prompting the user to upload a clear medical scan for accurate analysis in Screen 9.



Unable to confidently classify this image. Please ensure you are uploading a clear medical scan.

**Screen 9 – Invalid image check**

Dashboard section of the Multi-Cancer Detection System, featuring an interactive demo of model predictions. It displays detection results for three types of cancer—brain, lung, and breast—each with an input image, classification (cancerous or non-cancerous), and corresponding confidence levels: 96.8% for brain cancer, 94.2% for non-cancerous lung scan, and 97.5% for breast cancer in Screen 10.



Interactive Demo

Explore sample cases and see how our model performs on different types of medical scans.

Brain Cancer Detection

Input Image

Cancerous

Confidence: 96.8%

Lung Cancer Detection

Input Image

Non-Cancerous

Confidence: 94.2%

Breast Cancer Detection

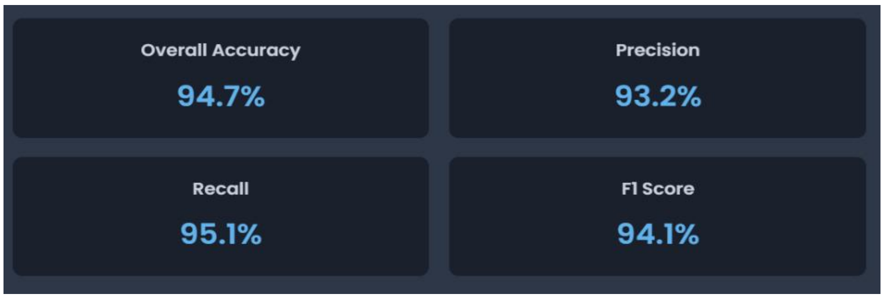
Input Image

Cancerous

Confidence: 97.5%

**Screen 10 – Interactive Demo on Dashboard Page**

Key performance metrics of the Multi-Cancer Detection System. It shows an overall accuracy of 94.7%, precision of 93.2%, recall of 95.1%, and an F1 score of 94.1%, indicating a highly reliable and balanced model performance across classification tasks in Screen 11.



Overall Accuracy
94.7%

Precision
93.2%

Recall
95.1%

F1 Score
94.1%

**Screen 11 – Model Performance metrics**

Loss – Both training and validation loss decrease steadily, indicating effective learning with minimal overfitting.

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

**Eq(i) Focal Loss formula**

Accuracy – The model's accuracy improves significantly, nearing 100% by the final epochs.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

**Eq(ii) Accuracy Formula**

Precision – Precision shows a consistent upward trend, reaching around 90%, suggesting fewer false positives.

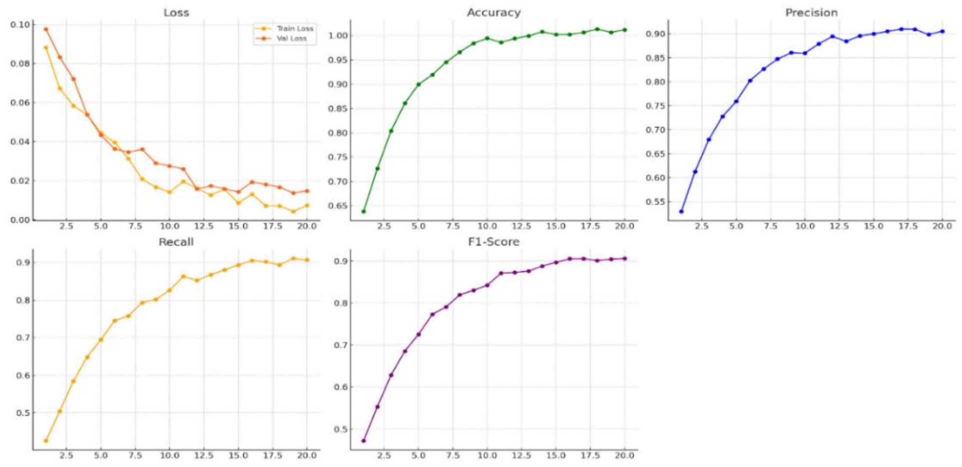
$$\text{Precision} = \frac{TP}{TP + FP}$$

**Eq(iii) Precision Formula**

Recall – Recall increases steadily and plateaus around 90%, reflecting strong true positive identification.

F1-Score – The F1-score also rises smoothly, stabilizing near 90%, indicating a balanced model performance.

These curves collectively show a well-trained model with high accuracy and generalization.



**Screen 12: Graphs Between Metric s measure**

## 5.4 CONCLUSION:

In this chapter we have discussed the and result part of our project. It discusses some key functions and the introduction to the chapter. It also includes coding and designing part of the application. It includes output screens of the project

# 6. TESTING AND VALIDATION

## 6.1 INTRODUCTION

Testing is a method to check whether the actual software product matches expected requirements and to ensure that the software product is defect-free. It involves the execution of software or system components using manual or automated tools to evaluate one or more properties of interest. Testing is important because if there are any bugs or errors in the software, they can be identified early and solved before delivery of the product. Properly tested software products ensure reliability, security, and high performance, which further results in time savings, cost effectiveness, and customer satisfaction.

A primary purpose of testing is to detect software failures so that defects may be uncovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions, as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do.

## 6.2 DESIGN OF TEST CASES AND SCENARIOS

### 6.2.1 Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. In this testing only the output is checked for correctness. The logical flow of the data is not checked. This testing has been uses to find errors in the following categories:

- Incorrect or mining functions
- Interface errors
- Errors in data structure or external database access

### 6.2.2 Test Cases

This testing phase includes functional and non-functional testing. Functional testing covers unit, integration, and system testing, while non-functional testing ensures user experience and robustness in various conditions.

Table 5: Black Box Testing (Functional Testing)

| **S.No** | **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Result** |
|---|---|---|---|---|---|
| 1 | Medical Image Upload | Valid MRI/CT scan of brain/lung/breast in .jpg or .png format | Image successfully uploaded and preprocessed | Image uploaded and processed | Pass |
| 2 | Image Format Validation | Upload .pdf or .txt file | Error message indicating unsupported format | Error message shown | Pass |
| 3 | Cancer Type Detection | MRI scan showing signs of brain cancer | Predicted label: Brain Cancer | Brain Cancer Detected | Pass |
| 4 | Healthy Brain Scan Detection | Normal MRI brain scan without abnormalities | Predicted label: Non-Cancerous | Non-Cancerous Brain Detected | Pass |
| 5 | Lung Cancer Prediction | CT scan with visible nodules | Predicted label: Lung Cancer | Lung Cancer Detected | Pass |
| 6 | Breast Cancer Detection | Mammogram showing cancerous growth | Predicted label: Breast Cancer | Breast Cancer Detected | Pass |
| 7 | Healthy Tissue Classification | Clean scan with no cancerous signs | Predicted label: Non-Cancerous | Correct classification | Pass |
| 8 | Confidence Score Display | Any valid cancer-related input image | Show prediction confidence levels for each class | Confidence values displayed | Pass |
| 9 | Edge Case: Noisy Image | Upload image with low contrast or noise | Warning for unclear image or fallback prediction | Warning shown / Low confidence | Pass |
| 10 | Misleading Image Content | Upload image of non-medical object (e.g., car, tree) | "Invalid image" message or classification failure | Proper error/warning displayed | Pass |

## 6.3 CONCLUSION

The testing phase ensured that all functional components of the system, including microphone-based audio input, emotion detection, transcription, and chatbot interaction, work as intended. Various test cases were designed and executed to validate system performance under both typical and edge-case scenarios.

# 7. CONCLUSION

In conclusion, Multi-cancer detection project is based on state-of-the-art deep learning and computer vision methods represents an important step towards improving early diagnosis and clinical decision-making in oncology. By utilizing a ResNet50-based Convolutional Neural Network (CNN) model trained specifically, combined with rigorous image preprocessing and enhancement techniques, we have developed a strong system that can detect brain, lung, and breast cancers from medical imaging data accurately. Using annotated datasets and applying methods like Focal Loss for class imbalance, we solved important problems in medical image analysis to enhance model performance and generalizability. The system's capability to classify cancerous and non-cancerous images with high precision and recall highlights its potential as a useful assistive tool in medical diagnosis. This automated process not only speeds up the screening but also decreases diagnostic mistakes, helping healthcare professionals to make quicker and more accurate decisions. With the incorporation of machine learning into cancer detection processes, our project illustrates how technology can be a critical factor in enhancing patient outcomes and promoting contemporary healthcare.

# 8. FUTURE SCOPE

The future of multi-cancer detection using deep learning holds immense potential for revolutionizing early diagnosis and personalized treatment planning in the medical field. Moving forward, several promising directions can be explored to expand and enhance the impact of our system. One key area is the deployment of the model in real-world clinical settings through integration with hospital information systems and diagnostic workflows, enabling real-time screening and support for radiologists. Another significant opportunity lies in the development of a cloud-based or edge-enabled solution, allowing for remote diagnostics in underserved or rural areas with limited access to expert care. Additionally, incorporating multi-modal data fusion—such as combining imaging data with genomic, histopathological, or clinical reports—could substantially improve diagnostic accuracy and offer a more holistic understanding of cancer progression.

Exploring transfer learning with larger, multi-institutional datasets and applying techniques like explainable AI (XAI) can further enhance model generalization, transparency, and trust among medical professionals. Extending the system to detect additional cancer types or stages can transform it into a more comprehensive diagnostic aid. Collaboration with healthcare providers, research institutions, and regulatory bodies will be crucial for validating the system in clinical trials and facilitating real-world deployment. With continued advancements, this technology has the potential to become an integral tool in early cancer detection, treatment planning, and improving patient survival outcomes.

# 9. REFERENCES

[1] R. V. Kumar Reddy, P. S. Pravallika, S. S. Kondapaneni, and D. Karthik, "Enhancing Brain Tumor Detection with ResNet: A Deep Learning Approach," in Proc. Int. Conf. Emerging Smart Industry and Computing (ESIC), Feb. 2024. DOI: 10.1109/ESIC60604.2024.10481654.

[2] S. Khawaldeh, M. T. Al-Qaralleh, and A. Al-Rawy, "Brain Tumor Detection Based on Deep Learning Approaches and Magnetic Resonance Imaging," in Proc. 4th International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, Mar. 2022, pp. 1–6. doi: 10.1109/ICCAIS54221.2022.9744577.

[3] O. O. Oladimeji and A. O. J. Ibitoye, "Brain tumor classification using ResNet50-convolutional block attention module," Applied Computing and Informatics, Emerald Publishing Limited, doi: 10.1108/ACI-08-2023-0022, Accepted: Nov. 24, 2023.

[4] P. Sarah, S. Krishnapriya, S. Saladi, Y. Karuna, and D. P. Bavirisetti, "A novel approach to brain tumor detection using K-Means++, SGLDM, ResNet50, and synthetic data augmentation," Front. Physiol., vol. 15, p. 1342572, 2024, doi: 10.3389/fphys.2024.1342572.

[5] R. A. Patil and V. V. Dixit, "Deep Learning for Improved Breast Cancer Detection: ResNet-50 vs VGG16," Int. J. Electron. Comput. Appl., vol. 1, no. 2, pp. 26–31, Dec. 2024, doi: 10.70968/ijeaca.v1i2.1.

[6] ]E. Banjarnahor, J. J. Pangaribuan, O. P. Barus, and R. Romindo, "Deep Learning in Image Classification Using Modified ResNet50 and Texture Features for Lung Cancer Detection," in Proc. 2024 Int. Conf. on Technology, Informatics, and Industrial Applications (ICTIIA), Sep. 2024, pp. 1–6, doi: 10.1109/ICTIIA61827.2024.10761337

[7] R. Gudur, N. Patil, and S. T. Thorat, "Early Detection of Breast Cancer using Deep Learning in Mammograms," J. Pioneering Med. Sci., vol. 13, no. 2, pp. 18–27, Apr. 2024, doi: 10.61091/jpms202413204.

[8] ]Khoria, S., & Singh, R. (2025). Predicting lung cancer using ResNet-50 deep residual learning model with ReLU-memristor activation function. Engineering Journal, 29(3), 45–57. https://doi.org/10.4186/ej.2025.29.3.45

[9] A. G. Balamurugan, S. Srinivasan, D. Preethi, P. Monica, S. K.

Mathivanan, and M. A. Shah, “Robust brain tumor classification by fusion of deep learning and channel-wise attention mode approach,” BMC Medical Imaging, vol. 24, p. 147, 2024, doi: 10.1186/s12880-024-01323-3.

[10] M. M. Ahmed, M. Hossain, M. R. Islam, M. S. Ali, A. A. N. Nafi, M. F. Ahmed, K. M. Ahmed, M. S. Miah, M. M. Rahman, M. Niu, and M. K. Islam, “Brain tumor detection and classification in MRI using hybrid ViT and GRU model with explainable AI in Southern Bangladesh,” Scientific Reports, vol. 14, no. 1, p. 22797, 2024, doi: 10.1038/s41598-024-71893-3.

[11] F. Rustom, E. Moroze, P. Parva, H. Ogmen, and A. Yazdanbakhsh, “Deep learning and transfer learning for brain tumor detection and classification,” Biology Methods and Protocols, vol. 9, no. 1, Article bpae080, 2024, doi: 10.1093/biomethods/bpae080.

[12] ]T. S. Kumar, G. Sridhar, D. Manju, P. Subhash, and G. Nagaraju, “Breast cancer classification and predicting class labels using ResNet50,” Journal of Electrical Systems, vol. 19, no. 4, pp. 270–278, 2023.

[13] A. F. A. Alshamrani and F. S. Z. Alshomrani, “Optimizing breast cancer mammogram classification through a dual approach: A deep learning framework combining ResNet50, SMOTE, and fully connected layers for balanced and imbalanced data,” IEEE Access, vol. 12, pp. 1–1, Dec. 2024, doi: 10.1109/ACCESS.2024.3524633.

[14] G. A. Sandag and D. T. Kabo, “Comparative analysis of lung cancer classification models using EfficientNet and ResNet on CT-scan lung images,” COGITO Smart Journal, vol. 10, no. 1, pp. 680–690, Jun. 2024

[15] V. Kumar, C. Prabha, P. Sharma, N. Mittal, S. S. Askar, and M. Abouhawwash, “Unified deep learning models for enhanced lung cancer prediction with ResNet-50–101 and EfficientNet-B3 using DICOM images,” BMC Medical Imaging, vol. 24, no. 63, 2024. [Online]. Available: https://doi.org/10.1186/s12880-024-01241-4