

# **“PneumoTrack: Smart Pneumonia Detection & Treatment”**

Mini Project submitted in partial fulfilment of the requirements for the award of  
the degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

*Under the esteemed guidance of*

**Dr. G. Bindu Madhavi**

Associate Professor, CSE(AI&ML) Department

Submitted by

**BHUKYA VEERANNA      20R11A6607**

**MADANU SHALINI      20R11A6631**

**POTTIPAKA VIJAY      20R11A6645**



**Department of Computer Science and Engineering CSE(AI&ML)**

**Accredited by NBA**

**Geethanjali College of Engineering and Technology**

**(UGC Autonomous)**

**(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)**

**Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.**

**2023-2024**



**Geethanjali College of Engineering & Technology**

**(UGC Autonomous)**

(Affiliated to JNTUH, Approved by AICTE, New Delhi)  
Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

**Department of Computer Science and Engineering(AI&ML)**

---

### **CERTIFICATE**

This is to certify that the Mini Project Report entitled “**PneumoTrack: Smart Pneumonia Detection & Treatment**” is a bonafide work done and submitted by

**BHUKYA VEERANNA      20R11A6607**

**MADANU SHALINI      20R11A6631**

**POTTIPAKA VIJAY      20R11A6645**

during the academic year **2023 - 2024**, in partial fulfilment of requirement for the award of Bachelor of Technology degree in “**Computer Science and Engineering (AI&ML)**” from Jawaharlal Nehru Technological University Hyderabad, is a bonafide record of work carried out by them under my guidance and supervision.

Certified further that to my best of the knowledge, the work in this dissertation has not been submitted to any other institution for the award of any degree or diploma.

**Project Guide**

**Dr. G. Bindu Madhavi**

**Associate Professor CSE(AI&ML)**

**Project Co-Ordinator**

**Mr. Shaik Akbar**

**Associate Professor CSE(AI&ML)**

**Head of the Department CSE(AI&ML)**

**Dr. L. Venkateswarlu**

**External Examiner**

## DECLARATION

We hereby declare that the Mini Project report entitled “**PneumoTrack: Smart Pneumonia Detection & Treatment**” is an original work done and submitted to **Computer Science and Engineering (AI&ML)** Department, **Geethanjali College of Engineering & Technology**, affiliated to Jawaharlal Nehru Technological University Hyderabad, in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering (AI&ML) and it is a record of bonafide project work carried out by us under the guidance of **Dr. G. Bindu Madhavi**, Associate Professor, Department of Computer Science and Engineering (AI&ML).

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma.

---

Signature of the Student

Bhukya Veeranna

20R11A6607

---

Signature of the Student

Madanu Shalini

20R11A6631

---

Signature of the Student

Pottipaka Vijay

20R11A6645

## ACKNOWLEDGEMENT

The satisfaction of completing this project would be incomplete without mentioning our gratitude towards all the people who have supported us. Constant guidance and encouragement have been instrumental in the completion of this project.

First and foremost, we thank the Chairman, Principal, Vice Principal for availing infrastructural facilities to complete the major project in time.

We offer our sincere gratitude to our Project guide **Dr. G. Bindu Madhavi**, Associate Professor, Computer Science and Engineering (AI&ML) Department, Geethanjali College of Engineering & Technology for her immense support, timely co- operation and valuable advice throughout the course of our Mini Project work.

We would like to thank the Project Co-Ordinator, **Mr. Shaik Akbar**, Associate Professor, Computer Science and Engineering (AI&ML) for his valuable suggestions in implementing the project.

We would like to thank the Head of Department CSE(AI&ML), **Dr. L. Venkateswarlu**, for his meticulous care and cooperation throughout the project work.

<b>BHUKYA VEERANNA</b>	<b>20R11A6607</b>
<b>MADANU SHALINI</b>	<b>20R11A6631</b>
<b>POTTIPAKA VIJAY</b>	<b>20R11A6645</b>

## ABSTRACT

Pneumonia is a life-threatening infectious disease that accounts for one in three in India. To address this challenge, we propose an advanced pneumonia detection and personalized treatment recommendation system. The system features a symptom-based prediction layer to assess pneumonia severity based on user-reported symptoms, enabling tailored initial treatment suggestions or further evaluation through X-ray images scans. Additionally, the system employs deep learning models to analyze chest X-ray images and classify the type of pneumonia, providing personalized treatment recommendations, including medications, diet plans, and exercise guidelines. Implementing a user-friendly application, the system empowers individuals to self-pneumonia management. However, we emphasize the importance of consulting healthcare professionals for accurate diagnosis and personalized treatment planning, ensuring the application serves as a valuable supportive tool in the overall healthcare process.

***Keywords: Pneumonia, Infectious, Symptom-based prediction, Deep Learning, Classification, Medication, accurate diagnosis, user-friendly application.***

## LIST OF FIGURES

Fig. 1.1(a) Non – Pneumonia Chest X-ray .....	1
Fig. 1.1(b) Pneumonia Chest X-ray .....	1
Fig. 1.2.1 Causes of Pneumonia .....	2
Fig. 1.4.1 Symptoms of Pneumonia .....	4
Fig. 4.3.1 Python logo.....	18
Fig. 4.3.2 TensorFlow logo .....	18
Fig. 4.3.3 scikit learn logo .....	19
Fig. 4.3.4 SQLite logo.....	19
Fig.4.3.5 VS Code logo .....	20
Fig. 4.3.6 jupyter, colab logo .....	20
Fig. 4.3.7 PyQt designer logo.....	21
Fig. 5.1.1 Process Cycle.....	23
Fig. 5.2.1 User Flow Diagram.....	24
Fig. 5.3.1 E R Diagram .....	25
Fig. 5.3.2 Schema in SQLInte3 .....	25
Fig. 6.2.1 efficientNetB0 Architecture .....	27
Fig. 6.2.2 Internal Process of Dense Net.....	27
Fig. 6.3.1 Model Architecture .....	28
Fig. 6.4.1 Internal Implementation.....	29
Fig. 7.1.1 process and training of pre-trained model.....	32
Fig. 7.2.1 Training for severity prediction .....	33
Fig. 7.2.2 Training for Disease prediction.....	33
Fig. 7.2.3 Training for medicine recommendations .....	33
Fig. 7.3.1 find accuracy for severity prediction .....	34
Fig. 7.3.2 Best epoch value for Disease prediction .....	35
Fig. 7.3.3 confusion matrix .....	35
Fig. 7.4.1 Evaluation results .....	36
Fig. 10.1 GUI Home Page .....	43
Fig. 10.2 Login Page.....	44
Fig. 10.3 Signup Page.....	45
Fig. 10.4 Options Page .....	46
Fig. 10.5 Severity Prediction Page .....	47
Fig. 10.6 Disease Prediction Pahe .....	48
Fig. 10.7 Recommendations Page .....	49

# TABLE OF CONTENTS

**Certificate Page**

**Declaration**

**Acknowledgement**

**Abstract .....i**

**List of Figures .....ii**

**Table of Contents .....iii - v**

## **1. INTRODUCTION**

1.1 About Pneumonia .....	1
1.2 Causes of Pneumonia .....	2
1.3 Risk Factors .....	3
1.4 Symptoms of Pneumonia .....	4
1.5 Existing System.....	5
1.6 Limitations of Existing Systems .....	6-7
1.7 Proposed System.....	8
1.8 Advantages of Proposed System.....	8

## **2. AIM and OBJECTIVES**

2.1 AIM of proposed system.....	9
2.2 Objectives of proposed system.....	10-11

## **3. LITERATURE SURVEY**

3.1 Introduction.....	12
3.2 Pneumonia Detection using various Techniques .....	12-15

## **4. SOFTWARE / HARDWARE REQUIREMENTS**

4.1 Functional Requirements .....	16
4.2 Non – Functional Requirements .....	17
4.3 Software Requirements .....	18-21
4.4 Hardware Requirements.....	22
4.5 Tech stack .....	22

## **5. SOFTWARE DESIGN**

5.1 process Cycle.....	23
5.2 User Flow Diagram .....	24
5.3 E R Diagram .....	24-25

## **6.SYSTEM IMPLEMENTATION**

6.1 Methods.....	26
6.2 Model.....	27
6.3 Model Implementation .....	28
6.4 Internal Implementation.....	28-31

## **7. TRAINING and TESTING**

7.1 Training .....	32
7.2 Model Fit and epochs .....	33
7.3 Testing and Evaluation.....	34-35
7.4 Evaluation .....	36

## **8. USER INTERFACE INTEGRATION**

8.1 PyQt Designer Interface.....	37
----------------------------------	----

## **9. SAMPLE CODE**

9.1 level-1.ipynb .....	38-40
9.2 level-2.ipynb .....	41
9.3 level-3.ipynb .....	42

## **10. OUTPUT SCREENS**

10.1 GUI Home Page.....	43
10.2 Login Page .....	44
10.3 Signup Page .....	45
10.4 Options Page.....	46
10.5 Severity Prediction Page.....	47



10.6 Disease Prediction Page.....	48
10.7 Recommendations Page.....	49
<b>11. CONCLUSION .....</b>	<b>50</b>
<b>12. FUTURE SCOPE.....</b>	<b>51</b>
<b>13. BIBILOGRAPHY .....</b>	<b>52-53</b>

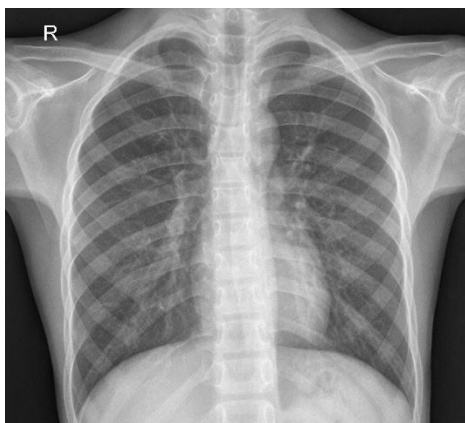
# 1. INTRODUCTION

## 1.1 About Pneumonia

7% of the population or 450 million people worldwide are suffering from Pneumonia which results in the death of 4 million people every year. Pneumonia is an acute lower respiratory disease with traces of infection usually because of some micro-organism, bacteria or virus. World health Organization (WHO) states that the pneumonia is the leading reason for the child death in the World. It has killed approximately 1.2 million children under the age of five.

South Asian countries like Indian Pakistan, Bangladesh, Sri Lanka, Indonesia tec. have the most no. children having pneumonia disease like AIDS, malaria and tuberculosis combined. Among children younger than five years, pneumonia is considered as one of the deadliest disease. Only India has encountered 158,176 deaths in the year 2016, and still India continues to show the significant no. in infant deaths because of pneumonia.

Although, some studies tell us that discrepancies are usual in the interpretation of the chest radiographs via radiologist. This drawback of human based observation has provided enough motivation to technology making interferences in this field where machine will classify between a normal X-ray and abnormal chest X-ray. This provides accuracy and diagnosis process.



**Fig. 1.1(a) Non-Pneumonia Chest X-ray**



**Fig.1.1(b) Pneumonia Chest X-ray**

## 1.2 Causes of Pneumonia

### 1. Bacteria

Bacteria are a common cause of pneumonia in adults. Many tu[es can cause pneumonia, but “**Streptococcus**” pneumonia is the most common. Some bacteria can cause pneumonia with different symptoms or other characteristics than the usual pneumonia.

### 2. Viruses

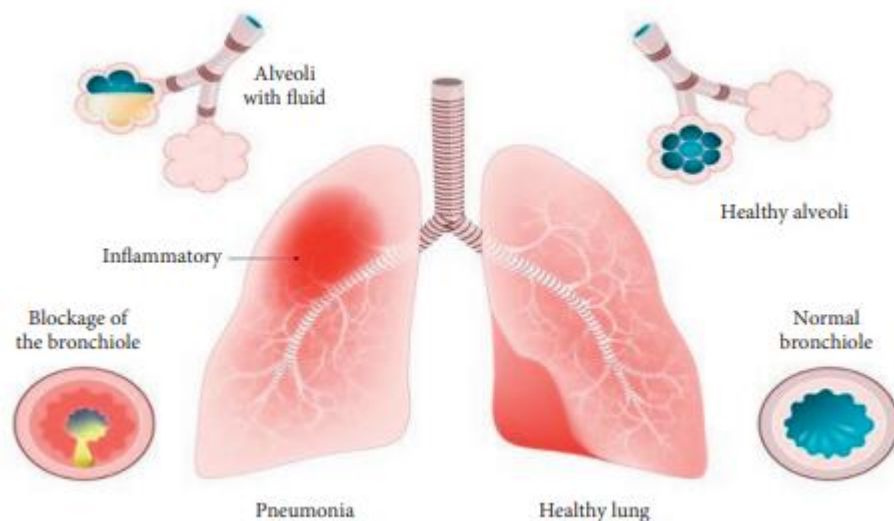
Viruses that infect lungs and airways can cause pneumonia. The flu and common cold are the most common causes of viral pneumonia in adults. Respiratory Syncytial Virus (RSV) is the most common cause of viral pneumonia in young children.

### 3. Fungi

Fungi such as “**Pneumocystis jirovecii**” may cause pneumonia, especially for people who have weakened immune systems. Some fungi found in the soil in southwestern United States can also cause pneumonia.

Some are the common disease that can leads to pneumonia are:

- Common cold
- Flu (Influenza Virus)
- COVID-19



**Fig.1.2.1 Causes of Pneumonia**

## 1.3 Risk Factors

### 1. Age

Pneumonia can affect people of all ages. However, two age groups are at higher risk of developing pneumonia and having more serious pneumonia.

- **Babies and children (2 years old or younger)**

They are at higher risk because their immune systems are still developing. This risk is higher for premature babies.

- **Older adults (age 65 or older )**

Are at higher risk because their immune systems generally weaken as people age. Older adults are also more likely to have chronic health conditions that raise the risk of pneumonia.

### 2. Environment or Occupation

Most people get pneumonia when they catch an infection from someone else in their community. Chance of getting pneumonia is higher if we live or spend a lot of time in a crowded place such as military barracks, prison, homeless shelters, or nursing homes. Some germs that cause pneumonia can infect birds and other animals. These germs can encounter if we work in a chicken processing center, pet shop or veterinary clinic.

### 3. Lifestyle habits

- Smoking cigarettes can make you less able to clear mucus from your airways.
- Using drugs or alcohol can weaken immune systems. We are also more likely to accidentally breathe in saliva or vomit into our windpipe from an overdose.

### 4. Other medical conditions

- Brain disorders such as a stroke, a head injury, dementia can affect ability to cough. This can lead to food or liquid going down windpipe instead of esophagus and getting into lungs.
- Critical diseases that require hospitalization including receiving treatment in a hospital intensive care unit, can raise risk of hospital-acquired pneumonia.
- Other serious conditions such as malnutrition, diabetes, heart failure, sickle cell disease or liver or kidney disease are additional risk factors.

## 1.4 Symptoms of Pneumonia

The signs and symptoms of pneumonia vary from mild to severe, depending on factors such as the type of germ causing the infection, and your age and overall health. Mild signs and symptoms often are similar to those of a cold or flu, but they last longer.

Signs and symptoms of pneumonia may include:

- **Chest pain** when we breathe or cough
- **Confusion** or change in mental awareness
- **Cough**
- **Fatigue**
- **Fever, sweating and shaking chills**
- **Nausea, vomiting or diarrhea**
- **Shortness of breath**



Fig. 1.4.1 symptoms of pneumonia

## 1.5 Existing System

### 1. Radiological Imaging and Diagnosis Tools

Various software and systems were available for radiologists and healthcare professionals to assist in the interpretation of chest X-rays and CT scans. These tools could aid in identifying pneumonia patterns, but they typically required expert interpretation.

### 2. Electronic Health Records (EHR) Systems

EHR systems were widely used in healthcare to maintain patient records, including pneumonia diagnosis and treatment history. They helped streamline healthcare data management and access.

### 3. Telemedicine Platforms

Telemedicine applications and platforms allowed patients to consult with healthcare providers remotely. These platforms often included features for sharing symptoms and medical history, which could aid in pneumonia diagnosis.

### 4. Machine Learning and AI Tools

Various AI-powered tools and models were being developed for pneumonia detection from medical images. These tools aimed to automate the process and assist radiologists in identifying pneumonia-related abnormalities in X-rays and CT scans.

### 5. Clinical Decision Support Systems

These systems provided recommendations to healthcare providers based on patient data, including symptoms, test results, and medical history. Some could be adapted for pneumonia diagnosis and treatment planning.

### 6. Healthcare Mobile Apps

Several healthcare mobile applications provided general health information and symptom tracking features. Some might have included basic pneumonia-related content.

## 1.6 Limitations of Existing Systems

Keep in mind that the healthcare technology landscape is dynamic, and efforts are continuously made to address these limitations. Advances in AI, telemedicine, interoperability standards, and data security are ongoing, with the aim of improving healthcare delivery and patient outcomes. It's essential to consider these limitations and ongoing developments when evaluating and designing new healthcare systems.

### 1. Dependence on Expertise

Many existing systems relied heavily on expert interpretation, especially in the case of radiological imaging. Radiologists and healthcare professionals were often required to analyze X-rays and CT scans for pneumonia diagnosis, which could lead to delays and potential variations in diagnosis accuracy.

### 2. Limited Access to Healthcare

Access to healthcare systems, including telemedicine platforms, was often constrained by geographic and socioeconomic factors. This limited the reach of healthcare services, especially in rural or underserved areas.

### 3. Data Privacy and Security Concerns

Electronic Health Records (EHR) systems and telemedicine platforms faced challenges related to data privacy and security. Safeguarding sensitive patient information was a critical concern, and breaches could lead to serious consequences.

### 4. Interoperability Issues

Interoperability problems between different EHR systems and healthcare providers were common. Sharing patient data across different systems and facilities could be challenging, affecting the continuity of care.

### 5. Accuracy of AI Models

AI-powered tools for pneumonia detection were promising but still faced challenges in achieving high levels of accuracy, especially when dealing with complex cases or subtle abnormalities in medical images.

## **6. Lack of Personalization**

Some clinical decision support systems provided generic recommendations but lacked personalization based on individual patient factors. Tailoring treatment plans to each patient's unique circumstances was a challenge.

## **7. Resource Constraints**

In resource-limited settings, access to advanced technology and diagnostic tools, such as high-quality X-ray machines or CT scanners, was limited, affecting the quality of diagnosis and treatment.

## **8. Patient Engagement**

Healthcare mobile apps and patient portals often struggled to maintain patient engagement over time. Ensuring that patients consistently used these tools for symptom tracking or self-management was challenging.

## **9. Research and Clinical Validation**

Prototypes and experimental systems might lack comprehensive clinical validation and real-world testing. This could limit their adoption in clinical practice.

## **10. Regulatory and Ethical Challenges**

Compliance with healthcare regulations and ethical standards, such as patient consent and data handling, could be complex and varied across regions and healthcare systems.



## 1.7 Proposed System

The proposed system aims to enhance the detection and treatment of pneumonia, a life-threatening disease, with a focus on improving outcomes for children and the elderly in India. The system leverages cutting-edge technology, including Deep Convolutional Neural Networks (DCNNs) for feature extraction from chest X-ray images and traditional Machine Learning (ML) classification algorithms for accurate diagnosis. It builds upon the work of previous researchers who have successfully used DCNNs like VGG-16, AlexNet, Mask-RCNN, and others for pneumonia detection. Furthermore, it incorporates a three-step preprocessing technique to improve model generality.

By using DCNNs and ML classifiers, the system can distinguish between pneumonia patients and healthy individuals, facilitating early intervention and personalized treatment recommendations. This approach holds promise for improving healthcare outcomes and reducing the impact of pneumonia, particularly in vulnerable populations.

## 1.8 Advantages of Proposed System

**Table 1.8.1 Advantages of Proposed System**

Aspect	Proposed System	Existing Systems
<b>Pneumonia Detection Accuracy</b>	Utilizes DCNNs and ML for accurate diagnosis	Existing systems may rely on expert interpretation
<b>Personalized Treatment</b>	Provides tailored treatment recommendations	Limited personalization in existing systems
<b>Symptom-Based Prediction</b>	Utilizes user-reported symptoms for assessment	Some existing systems lack symptom-based prediction
<b>User-Friendly Application</b>	Offers an intuitive and accessible interface	Usability varies among existing healthcare apps
<b>Use of Pretrained Models</b>	Fine-tunes DCNNs with pretrained models	May not consistently leverage pretrained networks
<b>Impact on Vulnerable Populations</b>	Particularly beneficial for children and elderly	Generalized impact across demographic groups

## **2. AIM and OBJECTIVES**

### **2.1 AIM of proposed System**

The proposed system's primary objective is to address the significant challenge posed by pneumonia, a life-threatening infectious disease that has a substantial impact in India. At its core, this system aims to revolutionize pneumonia detection and treatment by leveraging advanced technology. It begins by utilizing a symptom-based prediction layer, which assesses the severity of pneumonia based on user-reported symptoms. This initial assessment enables the system to offer tailored treatment suggestions, providing a crucial first step in pneumonia management. Users are guided towards either self-management or further evaluation through chest X-ray scans, depending on the severity of their condition.

Furthermore, the system employs deep learning models to analyze chest X-ray images. By doing so, it not only confirms the presence of pneumonia but also classifies its specific type, which is crucial for determining the most effective treatment approach. The personalized treatment recommendations encompass a range of interventions, including medications, diet plans, and exercise guidelines, designed to cater to the unique needs of each individual. Importantly, the system is implemented through a user-friendly application, making it accessible to a wide audience, and empowering individuals to take an active role in managing their pneumonia.

However, it's essential to underscore that while this system offers valuable support and guidance, it does not replace the expertise of healthcare professionals. Consulting with healthcare professionals remains of utmost importance for obtaining an accurate diagnosis and developing a personalized treatment plan tailored to an individual's specific circumstances. The proposed system is intended to complement and enhance the overall healthcare process, ensuring that individuals receive the best possible care and guidance in managing pneumonia.

## 2.2 Objectives of proposed system

### 1. Develop a Symptom-Based Prediction Layer

The first objective is to create a symptom-based prediction layer within the system. This layer will enable the assessment of pneumonia severity based on user-reported symptoms. It involves developing algorithms or models that can analyze and interpret symptoms provided by users to make an initial assessment of the condition. This prediction layer serves as the entry point for users into the system and determines whether they should proceed with self-management suggestions or if further evaluation through X-ray scans is necessary. The development of this prediction layer should involve rigorous data analysis and model training to ensure its accuracy.

### 2. Implement Deep Learning Models for X-ray Analysis

The second objective is to incorporate deep learning models into the system for the analysis of chest X-ray images. These models will be designed to confirm the presence of pneumonia and classify its type. This involves creating or adopting state-of-the-art deep learning architectures for image analysis. The models should be trained on a comprehensive dataset of chest X-ray images, encompassing various pneumonia types and severities. The accuracy and efficiency of these models are crucial for providing reliable diagnostic information to users.

### 3. Provide Personalized Treatment Recommendations

The third objective focuses on delivering personalized treatment recommendations based on the assessment made by the system. Once a user's pneumonia severity and type have been determined, the system should generate tailored treatment suggestions. These recommendations should encompass medication prescriptions, dietary plans, and exercise guidelines. This requires the integration of medical knowledge and best practices into the system's algorithms to ensure that the provided recommendations are safe and effective for each individual's unique condition.

#### **4. Create a User-Friendly Application**

The fourth objective is to design and develop a user-friendly application that facilitates easy interaction with the system. The application should have an intuitive user interface, clear navigation, and user-friendly features. It should guide users through symptom reporting, X-ray image submission (if required), and present treatment recommendations in a comprehensible manner. The application's design and functionality should prioritize accessibility to a broad range of users, including those with limited technical expertise.

#### **5. Promote Early Intervention and Better Pneumonia Management**

The fifth and overarching objective is to promote early intervention and improve pneumonia management. This objective is the ultimate goal of the project, aiming to reduce the morbidity and mortality associated with pneumonia in India. Achieving this objective involves the successful implementation of the previous four objectives, ensuring that individuals receive timely and accurate guidance for managing their pneumonia. It also underscores the importance of consulting healthcare professionals for comprehensive diagnosis and personalized treatment planning while utilizing the proposed system as a valuable supportive tool.

## 3. LITERATURE SURVEY

### 3.1 Introduction

Pneumonia, a life-threatening infectious disease, poses a significant public health challenge, particularly in regions like India. Addressing this challenge requires innovative approaches that leverage advanced technology and medical insights. In the pursuit of enhancing pneumonia detection and treatment, it is essential to ground our efforts in a comprehensive understanding of existing research and developments in the field. This literature review serves as a foundational exploration of relevant studies, highlighting key findings, trends, and gaps that inform our project's objectives.

The literature review is structured to cover critical aspects of our project, including pneumonia detection, personalized treatment recommendations, user-friendly healthcare applications, and the role of healthcare professionals. Through a systematic analysis of prior research, we aim to provide a context that underscores the significance of our project's objectives and its potential impact on improving healthcare outcomes.

By examining the existing body of knowledge, we establish a solid foundation upon which to build our advanced pneumonia detection and personalized treatment recommendation system. This review enables us to align our project with established best practices, emerging technologies, and the imperative for early intervention, ensuring that our system can make a meaningful contribution to the healthcare landscape, particularly in the context of pneumonia management in India.

### 3.2 Pneumonia Detection using various Techniques

#### 1. Utilization of X-ray Images for Pneumonia Identification

- Researchers like Togaçar et al. employed X-ray images of lungs for pneumonia detection.
- Convolutional Neural Networks (CNNs) were used as feature extractors, leveraging existing models such as VGG-16 and AlexNet.

- Feature selection algorithms were applied to reduce the number of deep features extracted from X-ray images.
- Traditional Machine Learning (ML) classifiers like Decision Trees (DT), Linear Discriminant Analysis (LDA), and Linear Regression were used for diagnosis with successful results.

## **2. Deep Learning Approaches for Pneumonia Diagnosis**

- Liang and Zheng developed a Deep Learning (DL) framework for child pneumonia diagnosis using image datasets, achieving satisfactory outcomes.
- Jaiswal et al. proposed a DL-based classification model using Mask-RCNN, demonstrating robustness and effectiveness.
- Ge et al. explored both ML (SVM, KNN, DT) and DL (MLP, RNN) models, achieving promising accuracy levels for pneumonia prediction.
- Sirazitdinov et al. designed an automated system using CNNs like Mask R-CNN and RetinaNet on chest X-rays with satisfactory results.
- Behzadi-Khormouji implemented DL, specifically CNN, using pretrained models and a three-step preprocessing technique to enhance model generality.
- Bhandary et al. introduced a DL-based healthcare framework, employing modified AlexNet and SVM to classify chest X-rays into normal and abnormal classes.

## **3. The Role of Medical Imaging in Disease Identification**

- Medical imaging, including chest X-rays, plays a crucial role in identifying various diseases.
- The classification of medical images, particularly chest X-rays, is a vital task for disease diagnosis.
- This study focuses on fine-tuned versions of Deep Convolutional Neural Network (DCNN) architectures (e.g., CNN, AlexNet, SqueezeNet, VGG16, VGG19, Inception V3) for feature extraction.
- ML classification algorithms are employed to distinguish pneumonia patients from healthy individuals.

4. **Togaçar et al.:** Togaçar and colleagues focused their research on the critical task of pneumonia identification using chest X-ray images. They employed Convolutional Neural Networks (CNNs) as feature extractors, utilizing popular pre-trained models like VGG-16 and AlexNet. To streamline the extracted features, they applied feature selection algorithms. Importantly, they combined deep learning and classical machine learning by employing Decision Trees (DT), Linear Discriminant Analysis (LDA), and Linear Regression for pneumonia diagnosis. Their study yielded commendable results, highlighting the synergistic potential of deep learning and traditional ML methods in accurate pneumonia diagnosis.
5. **Liang and Zheng :**Liang and Zheng's study introduced a deep learning-based framework for the diagnosis of pneumonia in pediatric cases, leveraging image datasets. While specific details aren't provided, their research signifies the effectiveness of deep learning techniques in automating feature extraction from medical images and achieving satisfactory results in the diagnosis of pediatric pneumonia.
6. **Jaiswal et al.:** Jaiswal and collaborators proposed a sophisticated approach for pneumonia diagnosis using chest X-ray images. Their classification/detection model was based on the Mask-RCNN architecture, renowned for its prowess in instance segmentation and object detection tasks. This research demonstrated the robustness and effectiveness of Mask-RCNN in precisely detecting pneumonia in X-ray images, showcasing its potential for advanced medical image analysis.
7. **Ge et al.:** Ge et al. embarked on a comprehensive study that explored pneumonia disease prediction using both traditional machine learning (ML) and deep learning (DL) models. They incorporated classical ML classifiers like Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Decision Trees (DT), in addition to DL models such as Multilayer Perceptrons (MLP) and Recurrent Neural Networks (RNN). The study's promising results emphasized the flexibility of employing both ML and DL approaches in the prediction of pneumonia, offering options for various medical imaging scenarios.
8. **Sirazitdinov et al.:** In their research, Sirazitdinov and team proposed an automated system for forecasting pneumonia based on chest X-ray images using ML algorithms. They harnessed the power of two types of Convolutional Neural Networks (CNNs), namely Mask R-CNN and RetinaNet. Their study demonstrated that ML can be

effectively employed for pneumonia prediction in medical images, yielding satisfactory results that hold potential for clinical applications.

9. **Behzadi-Khormouji et al.:** Behzadi-Khormouji and colleagues presented a DL-based method for diagnosing pneumonia using CNNs and chest X-ray images. To boost accuracy, they incorporated a pretrained Deep CNN (DCNN) model from the ImageNet dataset and introduced a three-step preprocessing technique to enhance model generality. The research yielded good results, highlighting the significance of preprocessing steps and the advantages of using pretrained models in DL-based medical image analysis.
10. **Bhandary et al.:** Bhandary et al. introduced a comprehensive healthcare framework targeting the detection of cancer and pneumonia. They employed a modified AlexNet for classifying chest X-rays into normal and abnormal classes, complementing this with Support Vector Machines (SVM) for classification. The study also assessed the performance of pretrained DL transfer functions, specifically VGG16 and AlexNet. This work underscored the versatility of deep learning in healthcare diagnostics and the potential for combining both handcrafted and learned features to enhance accuracy in lung cancer diagnosis and pneumonia detection.
11. **Naydenova E, Tsanas A:** This paper, published in the Journal of the Royal Society Interface, explores the use of data mining techniques for the diagnosis of childhood pneumonia. The authors investigate the potential of data mining methods to analyze and extract meaningful patterns and information from clinical data related to childhood pneumonia cases. By leveraging data mining, the study aims to improve the accuracy and efficiency of diagnosing this respiratory condition in children. This research is significant in the context of healthcare data analysis and the potential application of data-driven approaches to improve diagnostic processes.
12. **Hao B, Sotudian S:** This paper, published in the journal eLife in 2020, focuses on the early prediction of level-of-care requirements in patients with COVID-19. Given the critical nature of the COVID-19 pandemic, accurate prediction of the care level needed for patients is essential for optimizing healthcare resource allocation. The authors employ data-driven methods to develop predictive models that can determine the severity of COVID-19 cases and guide healthcare providers in making informed decisions regarding patient care.



## **4. SOFTWARE / HARDWARE REQUIREMENT**

### **4.1 Functional Requirements**

#### **1. User Registration and Authentication**

- Users should be able to create accounts and log in securely.
- Differentiate between healthcare professionals and patients for tailored access.

#### **2. Symptom Reporting**

- Users should be able to report their symptoms accurately.
- The system should process and store symptom data for assessment.

#### **3. Pneumonia Severity Assessment**

- Implement a symptom-based prediction layer to assess pneumonia severity.
- Use machine learning algorithms to provide initial severity classification.

#### **4. Image Upload and Analysis**

- Allow users to upload chest X-ray images for analysis.
- Employ deep learning models for image classification and pneumonia identification.

#### **5. Personalized Treatment Recommendations**

Generate personalized treatment plans based on severity and pneumonia type.

Recommend medications, diet plans, and exercise guidelines tailored to each individual.

#### **6. User-Friendly Application**

- Design an intuitive and user-friendly mobile or web application.
- Ensure easy navigation, clear instructions, and user support features.

#### **7. Integration with Healthcare Professionals**

- Enable healthcare professionals to access patient data securely.
- Facilitate communication between users and healthcare providers when necessary.

#### **8. Data Privacy and Security**

- Implement robust data encryption and security measures to protect user information.
- Comply with healthcare data privacy regulations and standards.

## 4.2 Non – Functional Requirements

### 1. Performance

- The system should provide quick responses for symptom assessment and image analysis.
- It should handle concurrent user interactions efficiently.

### 2. Scalability

- Design the system to accommodate a growing user base and increasing data volume.
- Ensure scalability to meet potential future demands.

### 3. Accuracy

- Deep learning models should achieve high accuracy in pneumonia detection.
- Minimize false positives and false negatives in diagnosis.

### 4. Reliability

- Ensure system uptime and reliability for critical healthcare support.
- Implement backup and recovery mechanisms to prevent data loss.

### 5. Usability

- The user interface should be intuitive and accessible to a diverse user base.
- Consider usability testing and user feedback for continuous improvement.

### 6. Security

- Protect user data and medical records with robust security protocols.
- Regularly update security measures to address emerging threats.

### 7. Compliance

- Comply with relevant healthcare regulations, such as HIPAA in the United States or equivalent laws in other regions.
- Adhere to ethical standards in data handling and patient care.

### 8. Interoperability

- Ensure compatibility with existing healthcare systems and standards for data exchange.
- Facilitate seamless integration with other healthcare IT solutions.

## 4.3 Software Requirements

### 1. Python



**Fig, 4.3.1 Python logo**

Python is a widely-used high-level programming language known for its simplicity and readability. It features clean and easy-to-understand syntax, making it a favorite among developers. Python is interpreted, cross-platform, and comes with an extensive standard library, reducing the need for custom code. It supports dynamic typing and object-oriented programming principles and is highly interoperable with other languages. Python has a thriving community and offers versatility, making it suitable for web development, data analysis, machine learning, and more. Its open-source nature and ease of learning have contributed to its widespread adoption.

### 2. Tensorflow



**Fig.4.3.2 Tensorflow logo**

TensorFlow is an open-source machine learning framework developed by Google. It's designed to facilitate the creation, training, and deployment of machine learning models, particularly deep learning models. TensorFlow offers a flexible and scalable platform for building various AI applications, including image and speech recognition, natural language processing, and more. Its key features include a high-level API for quick model development (Keras), support

for both CPU and GPU acceleration, and a large ecosystem of pre-built models and tools. TensorFlow's versatility and robust community support make it a powerful tool for machine learning and artificial intelligence projects.

### 3. scikit learn



**Fig. 4.3.3 scikit learn logo**

Scikit-learn is a powerful machine learning library in Python that simplifies the process of building and applying machine learning models. It is widely used for tasks like classification, regression, clustering, and dimensionality reduction. Scikit-learn provides a user-friendly and consistent interface for various machine learning algorithms, making it an essential tool for data scientists and developers. It also offers comprehensive functionality for tasks like data preprocessing, model evaluation, and hyperparameter tuning. This library's simplicity, efficiency, and integration with other Python libraries make it a valuable asset in the field of machine learning and data analysis.

### 4. SQLite



**Fig. 4.3.4 SQLite logo**

SQLite is a lightweight, self-contained, and serverless relational database management system (RDBMS). It is widely used for embedded systems, mobile applications, and small-scale database solutions due to its minimal setup requirements and portability. SQLite stores its entire database as a single file, simplifying management and deployment. It supports SQL (Structured Query Language) for data manipulation and retrieval and provides transactional capabilities for

data consistency. SQLite is known for its efficiency and speed, making it a suitable choice for scenarios where a full-fledged RDBMS may be unnecessary or impractical.

## 5. VScode



**Fig. 4.3.5 VS code logo**

Visual Studio Code (VSCode) is a widely-used integrated development environment (IDE) known for its lightweight and versatile nature. It's developed by Microsoft and is favored by developers for its efficiency and extensibility. VSCode is highly customizable, allowing developers to tailor it to their specific needs with the help of numerous extensions available in its marketplace. Its key features include a powerful code editor with syntax highlighting, debugging support, and integrated version control.

## 6. Colab and Jupyter



**Fig. 4.3.6 jupyter, colab logo**

Jupyter is an open-source interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports various programming languages, including Python, R, and Julia. Jupyter notebooks, which are a key component of Jupyter, enable users to combine code execution with explanatory text, making it an excellent tool for data analysis, scientific research, and educational purposes.

Jupyter notebooks run in a web browser and can integrate with data visualization libraries, making it easier to explore and present data-driven insights.

Google Colab (Colaboratory) is a cloud-based platform provided by Google that allows users to create and run Jupyter notebooks without the need for local installation or powerful hardware. Colab offers free access to GPU and TPU resources, making it well-suited for machine learning and deep learning tasks. Users can collaborate on Colab notebooks in real-time, share them with others, and access pre-installed libraries and packages, simplifying the setup process for data analysis and machine learning projects. Colab notebooks are stored on Google Drive, ensuring easy access and version control.

## **7. PyQt Designer**



**Fig. 4.3.7 PyQt designer logo**

PyQt Designer is a visual tool for creating user interfaces (UIs) in PyQt, a set of Python bindings for the Qt application framework. PyQt Designer simplifies UI design by allowing developers to create GUI applications through a drag-and-drop interface. This tool generates XML-based UI files that can be loaded and utilized in PyQt applications. It streamlines the development process by enabling the design and layout of UI components, such as buttons, widgets, and dialogs, without the need for extensive manual coding. PyQt Designer enhances productivity, accelerates UI development, and is a valuable resource for developers creating PyQt-based applications.

## 4.4 Hardware Requirements

- **CPU:** A multi-core processor, preferably with support for parallel processing, is essential for running deep learning models efficiently. A modern CPU with multiple cores (e.g., quad-core or higher) is recommended.
- **RAM:** Adequate RAM is crucial, especially when dealing with large datasets and deep learning. Depending on your specific needs, a minimum of 16 GB of RAM is a good starting point, but more may be necessary for larger deployments.
- **GPU (Graphics Processing Unit):** To accelerate deep learning model training, consider using one or more GPUs. NVIDIA GPUs are commonly used for this purpose. High-end GPUs with CUDA support can significantly speed up the training of complex models, but we have used colab T4 Pro version of GPU.
- **Storage:** You will need sufficient storage capacity to store user data, medical records, deep learning model weights, and other system data. Consider fast SSDs for improved data access speed.
- **Compatibility:** Ensure that the user interface of your application is responsive and compatible with a variety of devices and screen sizes.
- **Network Bandwidth:** Adequate network bandwidth is essential, especially if your system handles a large number of image uploads and downloads. High-speed internet connectivity and sufficient internal network bandwidth are required for efficient data transfer.

## 4.5 Tech stack

- **Programming language** → Python
- **Development Framework** → Tensorflow, scikit – learn
- **Database** → SQLite
- **IDE** → VScode, colab, jupyter
- **Data visualization** → matplotlib, seaborn
- **Operating System** → Window 11
- **Model Development** → keras, EfficientNet
- **Interface Development** → PyQt Designer

## 5. SOFTWARE DESIGN

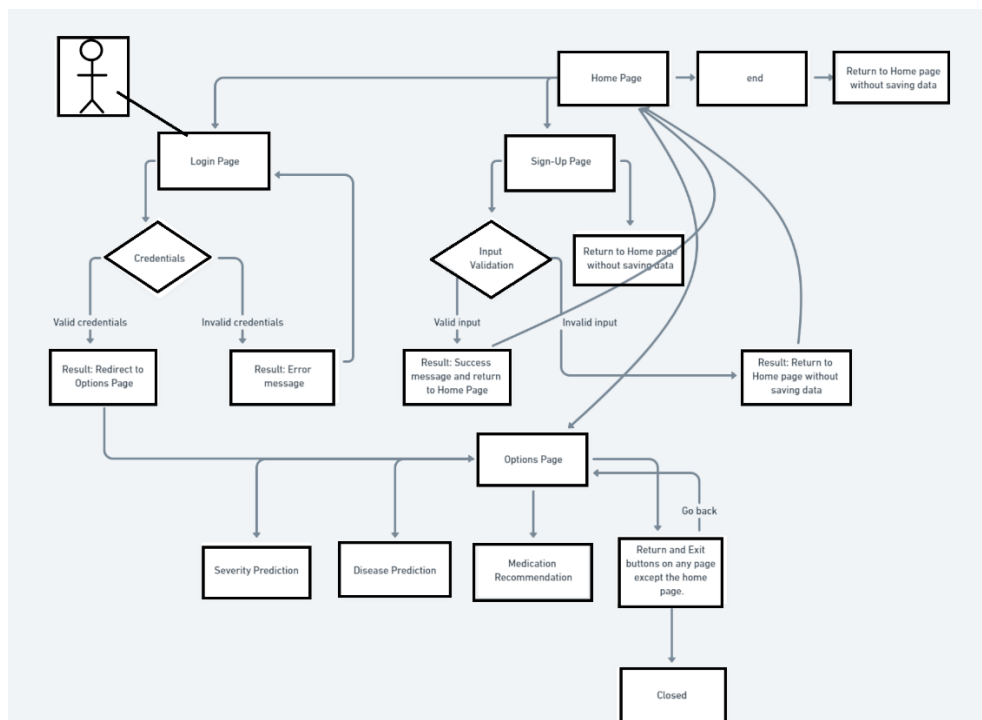
### 5.1 Process Cycle

The application begins with a Home Page where users can choose to "Login," "Sign Up," or "Exit." If they opt for "Login," they're directed to the Login Page to enter their credentials, and successful login leads to the Options Page. Registration occurs on the Sign-Up Page, with user data stored upon success.

The Options Page serves as the central hub, offering functionalities like Severity Prediction, Disease Prediction, and Medication Recommendation. Users can exit the app from here.

The Severity Prediction Page lets users input age and symptoms to predict pneumonia severity, with visual feedback based on results. Disease Prediction Page allows users to upload X-ray images for "Normal" or "Pneumonia" predictions.

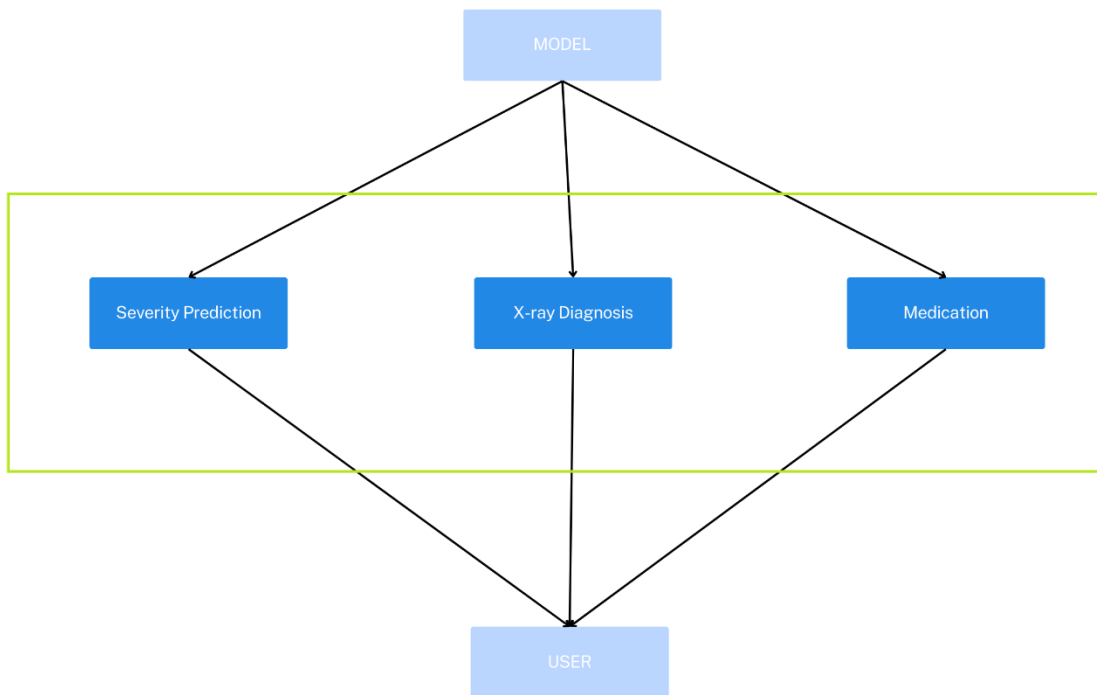
In the Medication Page, users select severity and diagnosis to receive medication recommendations. The process cycle enables users to navigate, interact, and make informed healthcare decisions regarding pneumonia prediction and treatment, all within the user-friendly graphical interface of the application.



**Fig. 5.1.1 Process cycle**



## 5.2 User Flow Diagram



**Fig. 5.2.1 User Flow Diagram**

## 5.3 E R Diagram

### users Table Schema Description:

1. id (INTEGER PRIMARY KEY AUTOINCREMENT): This field serves as the primary key for the table, ensuring each record has a unique identifier. The AUTOINCREMENT attribute automatically generates a unique ID for each new record.
2. name (TEXT NOT NULL): This field stores the user's full name as text. It's marked as "NOT NULL," meaning that a user's name must be provided and cannot be left empty.
3. age (INTEGER NOT NULL): This field stores the user's age as an integer. Similar to the "name" field, it's marked as "NOT NULL," indicating that age information must be provided.
4. gender (TEXT NOT NULL): This field stores the user's gender as text. Like the previous fields, it's marked as "NOT NULL," ensuring that gender information is required.

5. username (TEXT UNIQUE NOT NULL): This field stores the user's chosen username for authentication. It must be unique to each user, preventing duplicate usernames. It's marked as "NOT NULL" to ensure that a username is provided.
6. password (TEXT NOT NULL): This field stores the user's password for authentication. Like other sensitive data fields, it's marked as "NOT NULL."

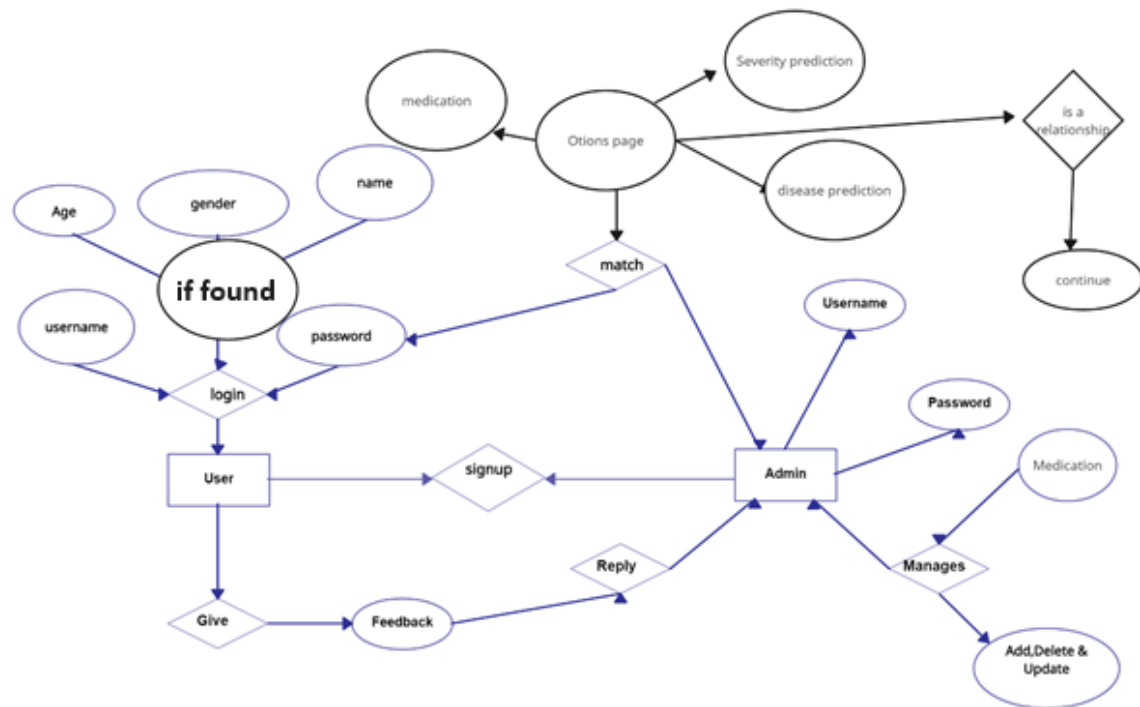


Fig. 5.3.1 E R Diagram

```
sqlite> .schema
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  age INTEGER NOT NULL,
  gender TEXT NOT NULL,
  username TEXT UNIQUE NOT NULL,
  password TEXT NOT NULL
);
CREATE TABLE sqlite_sequence(name,seq);
```

Fig. 5.3.2 Schema in SQLite3

## 6. SYSTEM IMPLEMENTATION

### 6.1 Methods

- **Data collection and preprocessing**

In this retrospective study, we enrolled adult patients aged 18 and above who had been diagnosed with acute lower respiratory illness and received treatment at the First Affiliated Hospital of Zhengzhou University in Henan Province. This hospital, one of the largest in central China with a capacity of approximately 13,000 beds, served as the research setting. Patient data encompassed demographic details such as age, gender, and comorbidities, as well as various physical parameters, including tachycardia, tracheal secretion, pleural effusion, mean arterial pressure, heart and breathing rates, and systolic blood pressure. Additionally, hematological parameters such as serum sodium, serum potassium, serum creatine, hematocrit, white blood cell (WBC) count, platelet count, total bilirubin, hemoglobin, C-reactive protein (CRP), and procalcitonin (PCT) were collected. To address missing data, which is common in real-world datasets, we undertook a data preprocessing stage consisting of four steps to enhance data quality and reliability for subsequent analysis.

- **Missing Values**

Missing data causes problems when a ML model is applied to the dataset. Mostly, ML models don't process data with missing values. In this study, some variables had several missing values of about 15% of that variable data. We used the median and mode of the corresponding columns to fill in the missing values of numerical attributes and categorical attributes, respectively. Median is the centrally located value of the dataset in ascending order. We filled missing numerical attribute values with the median value. Mode is the most repeated value in the given categorical observations. We filled missing entries with the mode observations.

- **Imbalance data**

The dataset used in this study exhibited a significant class imbalance, a common issue in classification datasets where one class greatly outweighs the other. This class with a larger proportion is referred to as the majority class, while the one with a smaller representation is the minority class. In this particular dataset, the imbalance was notable, with the minority class accounting for 22% fewer instances than the majority class. To

address this imbalance and reduce data bias before applying machine learning techniques, the Synthetic Minority Over-sampling Technique (SMOTE) was employed.

## 6.2 Model

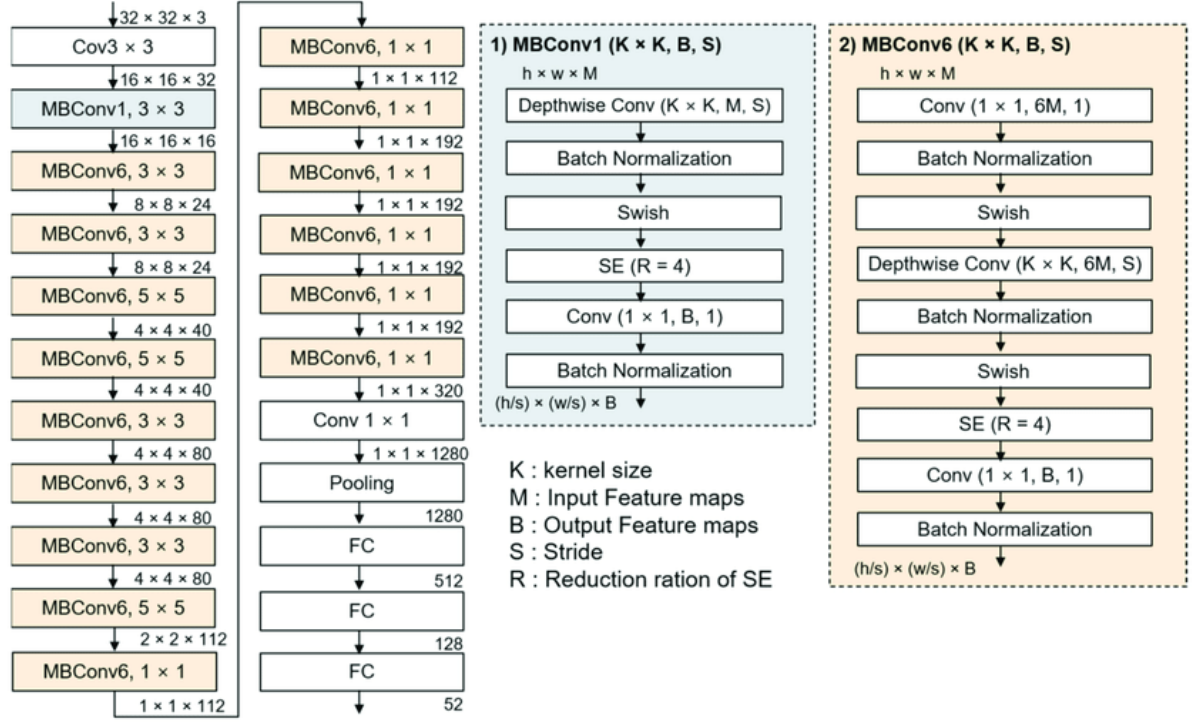


Fig. 6.2.1 efficientNetB0 Architecture

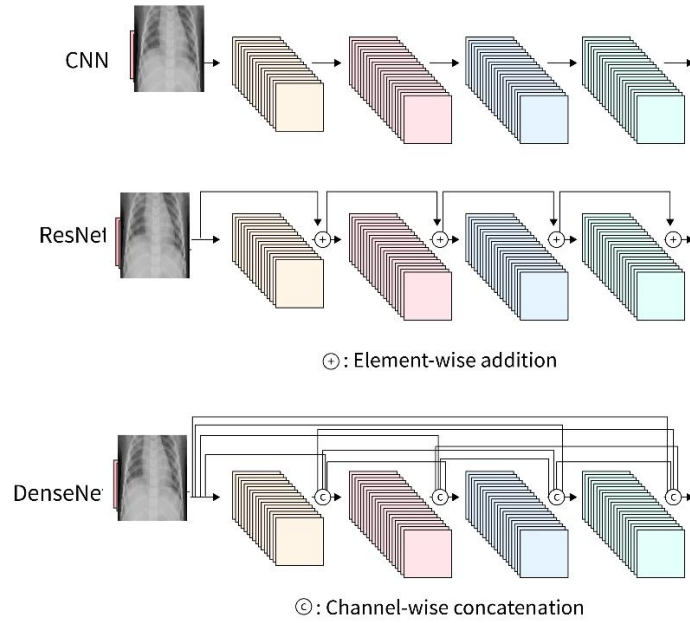


Fig. 6.2.2 Internal Process of Dense Net

## 6.3 Model Implementation

Implementing the EfficientNetB0 model internally requires a deep understanding of deep learning frameworks like TensorFlow or PyTorch, as well as access to the pre-trained model weights, unless you plan to train the model from scratch.

### Model Aechitecture

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
efficientnetb0 (Functional)	(None, 1280)	4049571
batch_normalization (Batch Normalization)	(None, 1280)	5120
dense (Dense)	(None, 256)	327936
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 2)	514

```
=====
Total params: 4383141 (16.72 MB)
Trainable params: 331010 (1.26 MB)
Non-trainable params: 4052131 (15.46 MB)
=====
```

Fig. 6.3.1 Model Architecture

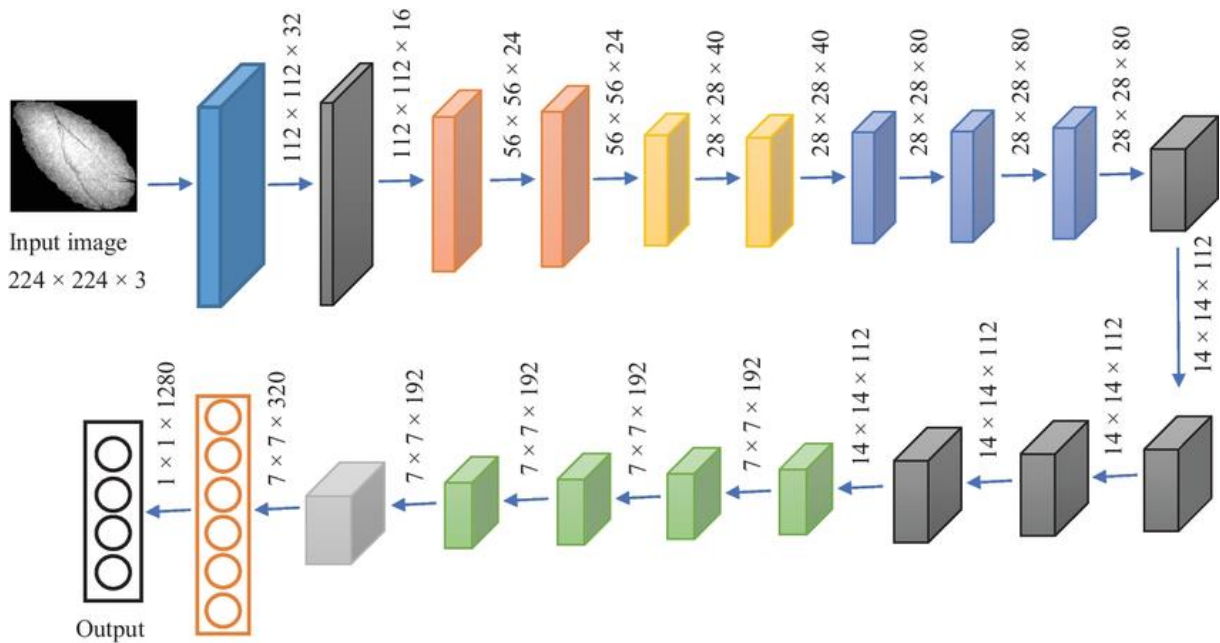
## 6.4 Internal Implementation

- **Compound Scaling:**EfficientNet models are scaled in multiple dimensions, including depth (number of layers), width (number of channels), and resolution (input image size). This compound scaling allows for efficient model design across different tasks and resource constraints.
- **Efficient Building Blocks:** EfficientNet models use novel building blocks called Mobile Inverted Residual Blocks (MBConv), which optimize the balance between

model depth and computational efficiency. These blocks incorporate depthwise separable convolutions to reduce parameters and computations.

- **Multi-Resolution Input:** EfficientNet models accept input images of different resolutions, allowing them to handle a wide range of image sizes effectively. This feature is particularly useful for applications with varying image resolutions.
- **State-of-the-Art Accuracy:** Despite its efficiency, EfficientNetB0 achieves state-of-the-art accuracy on various computer vision tasks, including image classification and object detection, when compared to other models with similar computational budgets.
- **Pre-trained Models:** Pre-trained versions of EfficientNetB0, along with other variants, are available in deep learning frameworks like TensorFlow and PyTorch. These pre-trained models can be fine-tuned for specific tasks with relatively small datasets.
- **Scalability:** The EfficientNet architecture is designed to be scalable, meaning that larger variants (e.g., B1, B2, B3, etc.) can be created by increasing the scaling factors. This allows users to choose models that match their hardware and accuracy requirements.

EfficientNetB0 and its variants have become popular choices in various computer vision applications due to their balance of model performance and computational efficiency, making them suitable for deployment in resource-constrained environments.



**Fig. 6.4.1 Internal Implementation**

## Components of efficientNetB0

EfficientNetB0, like other models in the EfficientNet family, is composed of several key components that contribute to its efficiency and effectiveness in deep learning tasks. These components include:

- **Input Layer:** EfficientNetB0 accepts images as input, typically with a fixed size, which can vary depending on the specific implementation but is often 224x224 pixels.
- **Convolutional Neural Network (CNN) Backbone:** The backbone of EfficientNetB0 consists of a series of convolutional layers organized into blocks. Each block contains multiple layers of depthwise separable convolutions, which are efficient in reducing computation while retaining representational power. These blocks help extract hierarchical features from the input images.
- **Depthwise Separable Convolutions:** EfficientNetB0 uses depthwise separable convolutions in its building blocks. These convolutions consist of a depthwise convolution followed by a pointwise convolution. Depthwise convolutions reduce the number of parameters by applying a separate convolution to each input channel, while pointwise convolutions combine the results.
- **Scaling Factors:** EfficientNet models are scaled in multiple dimensions, including depth, width, and resolution. Scaling factors are applied to these dimensions to control the model's size and computational requirements. EfficientNetB0 represents the smallest variant in the family.
- **Efficient Building Blocks:** Mobile Inverted Residual Blocks (MBConv) are the primary building blocks of EfficientNet models. These blocks are designed to optimize the trade-off between model depth and efficiency. They include depthwise separable convolutions and non-linear activations.
- **Feature Maps:** As the input image passes through the network, feature maps are generated at different levels of abstraction. These feature maps capture hierarchical information about the input image, from low-level details to high-level features.
- **Global Average Pooling (GAP):** EfficientNetB0 typically includes a Global Average Pooling layer at the end of the backbone. This layer reduces the spatial dimensions of the feature maps to a single vector, effectively summarizing the features extracted from the input image.

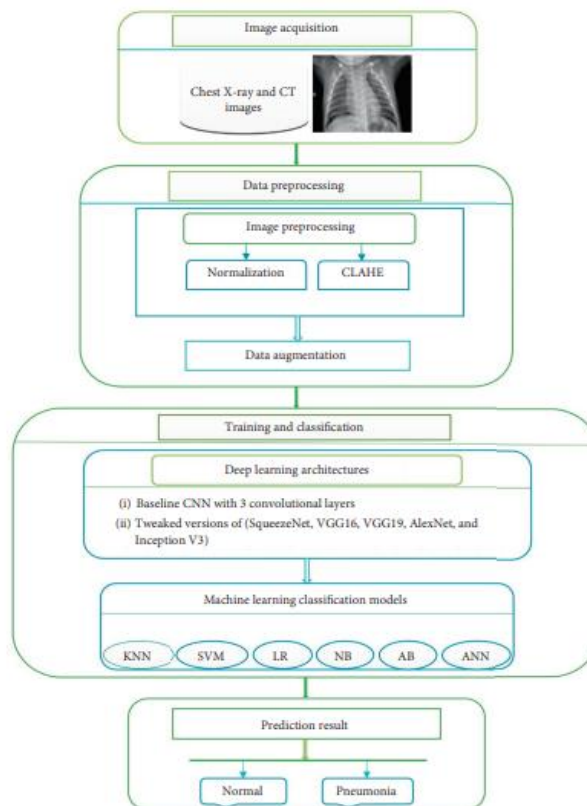
- **Fully Connected Layer (Classification Head):** Following the Global Average Pooling layer, EfficientNetB0 usually includes a fully connected layer (classification head) for making predictions. The number of output neurons in this layer corresponds to the number of classes in the classification task.
- **Activation Functions:** Activation functions, such as ReLU (Rectified Linear Unit) or Swish, are applied throughout the network to introduce non-linearity and enable the model to capture complex patterns in the data.
- **Dropout and Batch Normalization:** These regularization techniques may be applied to prevent overfitting and stabilize training during the learning process.



## 7. TRAINING and TESTING

### 7.1 Training

- Training a machine learning (ML) model that utilizes computer vision and deep learning involves the process of teaching a neural network to recognize and understand
- patterns, features, or objects within visual data, such as images or videos.
- Training an ML model that leverages computer vision and deep learning is an iterative and resource-intensive process that often requires expertise in machine learning,
- computer vision, and data engineering. Success in this field depends on the quality of data, model design, and the careful tuning of various components to achieve the desired results.
- Constructing a model necessitates a well-defined dataset split ratio. This project employed a 70:30 split ratio, where in 70 random images were selected for training
- from a pool of more than 5000 images for each pneumonia and rest is left for testing. This ensures a balanced and representative dataset for effective model training.



**Fig. 7.1.1 process and training of pre-trained model**

## 7.2 Model Fit and epochs

```
Epoch 5/50
1500/1500 [=====] - 4s 2ms/step - loss: 6.9088 - accuracy: 0.3312 - val_loss: 6.4190 - val_accuracy: 0.3348
Epoch 6/50
1500/1500 [=====] - 4s 3ms/step - loss: 4.3666 - accuracy: 0.3303 - val_loss: 3.4980 - val_accuracy: 0.3348
Epoch 7/50
1500/1500 [=====] - 6s 4ms/step - loss: 2.6381 - accuracy: 0.3302 - val_loss: 1.2031 - val_accuracy: 0.3348
Epoch 8/50
1500/1500 [=====] - 5s 3ms/step - loss: 2.0045 - accuracy: 0.3306 - val_loss: 0.2463 - val_accuracy: 0.3348
Epoch 9/50
1500/1500 [=====] - 3s 2ms/step - loss: 1.3558 - accuracy: 0.3305 - val_loss: 0.6578 - val_accuracy: 0.3348
Epoch 10/50
1500/1500 [=====] - 3s 2ms/step - loss: 0.1882 - accuracy: 0.3305 - val_loss: 0.1686 - val_accuracy: 0.3348
Epoch 11/50
1500/1500 [=====] - 3s 2ms/step - loss: 0.2543 - accuracy: 0.3305 - val_loss: 0.2711 - val_accuracy: 0.3348
...
Epoch 50/50
1500/1500 [=====] - 3s 2ms/step - loss: -8.8480 - accuracy: 0.3305 - val_loss: -1.9514 - val_accuracy: 0.3348
188/188 [=====] - 0s 2ms/step - loss: -1.9514 - accuracy: 0.3348
Neural Network Accuracy: 0.3348
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

**Fig. 7.2.1 Trainig for severity prediction**

```
Epoch 1/5
261/261 [=====] - 254s 932ms/step - loss: 5.1563 - accuracy: 0.8993 - val_loss: 3.3897 - val_accuracy: 0.9665
Epoch 2/5
261/261 [=====] - 267s 1s/step - loss: 2.6628 - accuracy: 0.9446 - val_loss: 1.9646 - val_accuracy: 0.9696
Epoch 3/5
261/261 [=====] - 287s 1s/step - loss: 1.5825 - accuracy: 0.9597 - val_loss: 1.2004 - val_accuracy: 0.9808
Epoch 4/5
261/261 [=====] - 279s 1s/step - loss: 1.0079 - accuracy: 0.9588 - val_loss: 0.8018 - val_accuracy: 0.9649
Epoch 5/5
261/261 [=====] - 293s 1s/step - loss: 0.6964 - accuracy: 0.9628 - val_loss: 0.5597 - val_accuracy: 0.9728
```

**Fig. 7.2.2 Training for Disease prediction**

```
Epoch 1/10
750/750 [=====] - 3s 3ms/step - loss: 2.2984 - accuracy: 0.5010 - val_loss: 0.8162 - val_accuracy: 0.5027
Epoch 2/10
750/750 [=====] - 2s 2ms/step - loss: 0.7644 - accuracy: 0.4958 - val_loss: 0.8254 - val_accuracy: 0.4992
Epoch 3/10
750/750 [=====] - 2s 2ms/step - loss: 0.7516 - accuracy: 0.5005 - val_loss: 0.7256 - val_accuracy: 0.5013
Epoch 4/10
750/750 [=====] - 2s 2ms/step - loss: 0.7390 - accuracy: 0.4988 - val_loss: 0.7943 - val_accuracy: 0.5002
Epoch 5/10
750/750 [=====] - 2s 2ms/step - loss: 0.7517 - accuracy: 0.5016 - val_loss: 0.7119 - val_accuracy: 0.5008
Epoch 6/10
750/750 [=====] - 2s 2ms/step - loss: 0.7435 - accuracy: 0.4980 - val_loss: 0.7409 - val_accuracy: 0.5102
Epoch 7/10
750/750 [=====] - 2s 2ms/step - loss: 0.7473 - accuracy: 0.5043 - val_loss: 0.7187 - val_accuracy: 0.4972
Epoch 8/10
750/750 [=====] - 2s 2ms/step - loss: 0.7436 - accuracy: 0.5028 - val_loss: 0.7268 - val_accuracy: 0.5095
Epoch 9/10
750/750 [=====] - 2s 2ms/step - loss: 0.7409 - accuracy: 0.5008 - val_loss: 0.7608 - val_accuracy: 0.5013
Epoch 10/10
750/750 [=====] - 2s 2ms/step - loss: 0.7279 - accuracy: 0.5036 - val_loss: 0.7259 - val_accuracy: 0.4937
```

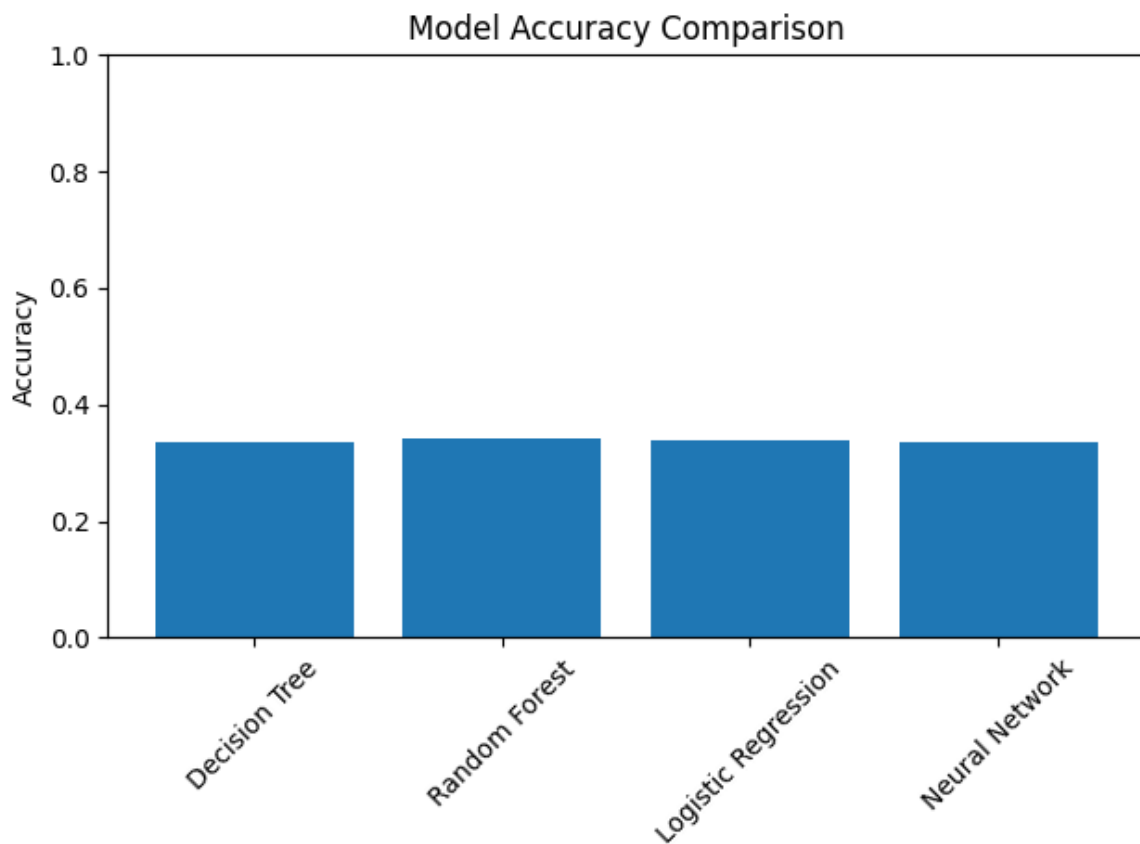
**Fig. 7.2.3 Training for medicine recommendation**

## 7.3 Testing and Evaluation

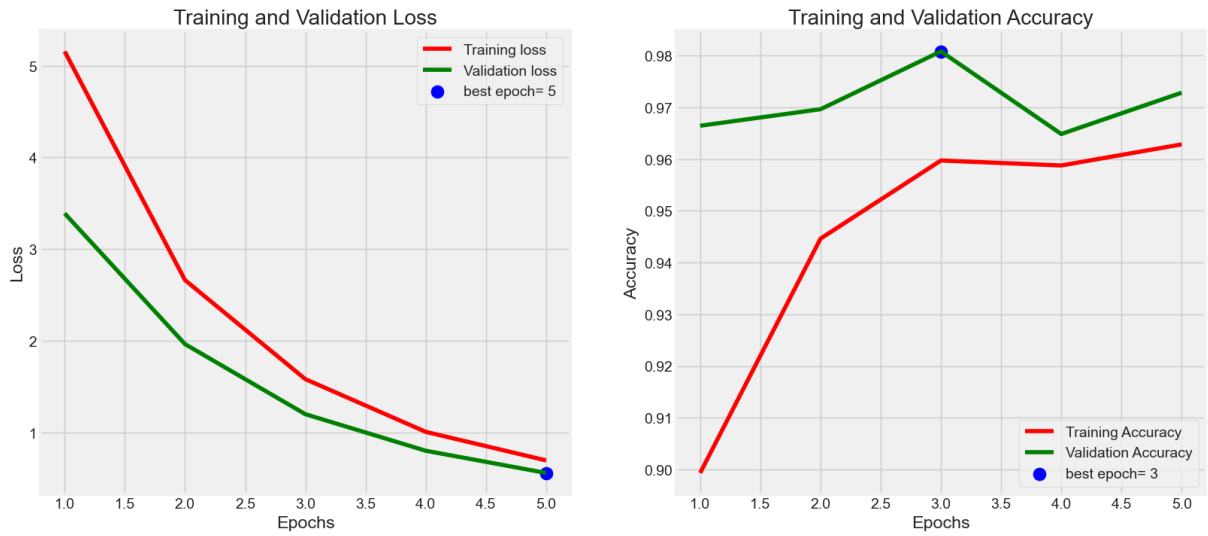
Testing and evaluation are essential phases in the development of machine learning models. These processes help assess a model's performance, reliability, and generalization ability. Here's a brief overview of testing and evaluation in machine learning:

### 7.3.1 Testing

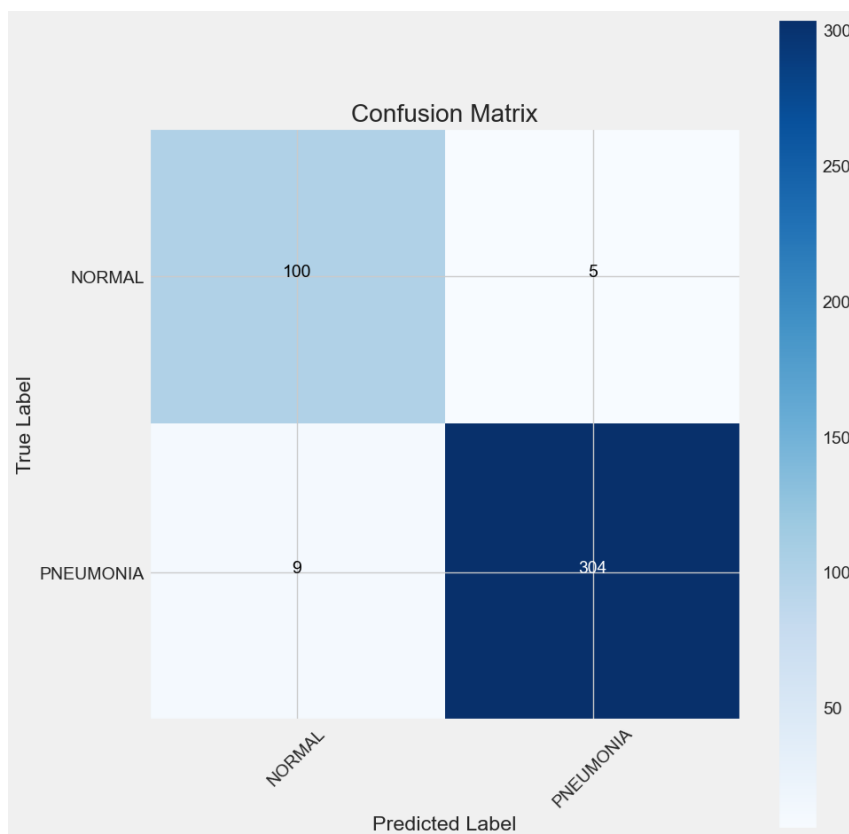
- **Data Splitting:** The dataset is typically divided into three sets: training, validation, and test sets. The training set is used to train the model, the validation set is used to fine-tune hyperparameters and monitor training progress, and the test set is used for unbiased model evaluation.
- **Model Training:** During training, the machine learning model learns from the training data. This involves optimizing model parameters (weights and biases) to make predictions that minimize a specified loss function.



**Fig. 7.3.1 final accuracy for severity prediction**



**Fig. 7.3.2 Best epoch value for Disease prediction**



**Fig. 7.3.3 confusion matrix**

## 7.4 Evaluation

1. **Performance Metrics:** Specific metrics are used to measure the model's performance. Common metrics include accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). The choice of metrics depends on the problem type (classification, regression, etc.) and goals.
2. **Model Validation:** The model's performance is assessed on the validation set. This helps in tuning hyperparameters and detecting issues like overfitting, where the model performs well on the training data but poorly on unseen data.
3. **Hyperparameter Tuning:** Hyperparameters (e.g., learning rate, batch size, model architecture) are adjusted to optimize model performance. Techniques like grid search or random search can be used to find the best combination of hyperparameters.
4. **Cross-Validation :** Cross-validation involves splitting the dataset into multiple subsets (folds) and training and evaluating the model on different combinations of these subsets. It provides a more robust estimate of model performance.
5. **Overfitting Analysis:** Evaluating whether the model is overfitting (learning noise in the data) or underfitting (not capturing important patterns) is crucial. Regularization techniques and proper model architecture can help mitigate overfitting.
6. **Final Model Evaluation:** The model's performance is assessed on the test set, which it has never seen before. This evaluation provides an unbiased estimate of the model's generalization ability and how well it will perform on new, unseen data.
7. **Baseline Comparison:** Models are often compared to baseline models or simple algorithms to gauge their effectiveness. A good machine learning model should outperform these baselines.

```
11/11 [=====] - 10s 892ms/step - loss: 0.5403 - accuracy: 0.9886
11/11 [=====] - 10s 897ms/step - loss: 0.5510 - accuracy: 0.9886
11/11 [=====] - 28s 3s/step - loss: 0.5716 - accuracy: 0.9665
Train Loss: 0.5403050780296326
Train Accuracy: 0.9886363744735718
-----
Validation Loss: 0.5509664416313171
Validation Accuracy: 0.9886363744735718
-----
Test Loss: 0.5716108679771423
Test Accuracy: 0.9665071964263916
```

Fig. 7.4.1 Evaluation results

## 8. USER INTERFACE INTEGRATION

### 8.1 PyQt Designer Interface

Integrating a user interface (UI) using PyQt involves creating a graphical interface for your application using the PyQt library. This process typically includes the following steps:

- **Installation:** Ensure that you have PyQt installed in your Python environment. You can install it using the pip package manager:

**`pip install PyQt5`**

- **UI Design:** Design your UI layout using Qt Designer, a visual interface design tool for the Qt framework. Qt Designer allows you to create a .ui file that defines the UI components, their positioning, and properties. Alternatively, you can design the UI programmatically in Python, but using Qt Designer is more common for complex layouts.
- **Conversion (If Using Designer):** If you design your UI using Qt Designer, you'll need to convert the .ui file into a Python script that PyQt can use. This conversion is typically done using the `pyuic` tool:

**`pyuic5 your_ui_file.ui -o ui_mainwindow.py`**

- **Application Script:** Write a Python script that imports PyQt modules and creates a PyQt application instance. This script also needs to instantiate your UI and connect it to your application's logic.
- **Logic Implementation:** Implement the application logic in your Python script. This includes event handling, data processing, and interactions with UI elements.
- **Binding UI Elements:** Connect UI elements (e.g., buttons, text fields, widgets) to corresponding functions or methods in your Python code. This step allows your UI to respond to user actions.
- **Execution:** Run the PyQt application by executing your Python script. This action launches the graphical user interface, allowing users to interact with your application.
- Throughout this process, you design the user interface to match your application's functionality and user experience requirements. PyQt provides extensive documentation and resources for creating sophisticated and interactive graphical user interfaces.

## 9. SAMPLE CODE

### 9.1 level-1.ipynb

```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MultiLabelBinarizer, LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
import matplotlib.pyplot as plt

# Module 1: Data Preprocessing and Model Training
def preprocess_data_and_train_models():
    # Load the dataset
    data = pd.read_csv('symptoms_Dataset.csv')

    # Data Preprocessing
    # Encode the 'Gender' column as binary
    data['Gender'] = data['Gender'].map({'Male': 0, 'Female': 1})

    # Convert the 'Symptoms' column to a binary matrix
    mlb = MultiLabelBinarizer()
    symptoms_encoded = mlb.fit_transform(data['Symptoms'].str.split(', '))
    symptoms_df = pd.DataFrame(symptoms_encoded, columns=mlb.classes_)

    # Combine the encoded 'Symptoms' with the original dataset
```

```

data = pd.concat([data, symptoms_df], axis=1)

# Drop the original 'Symptoms' column
data.drop('Symptoms', axis=1, inplace=True)

# Label Encoding for 'Severity'
label_encoder = LabelEncoder()
data['Severity'] = label_encoder.fit_transform(data['Severity'])

# Split the data into training and testing sets
X = data.drop('Severity', axis=1)
y = data['Severity']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Training
models = {
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Logistic Regression": LogisticRegression()
}

model_accuracies = {}

for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    model_accuracies[model_name] = accuracy

# Print Model Accuracies
for model_name, accuracy in model_accuracies.items():

```



```

print(f"{model_name} Accuracy: {accuracy:.4f}")

# Neural Network
nn_model = Sequential([
    Input(shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

nn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

nn_model.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_test, y_test))

_, nn_accuracy = nn_model.evaluate(X_test, y_test)
print(f"Neural Network Accuracy: {nn_accuracy:.4f}")

# Visualize Model Accuracies
model_accuracies["Neural Network"] = nn_accuracy

fig, ax = plt.subplots()
ax.bar(model_accuracies.keys(), model_accuracies.values())
ax.set_ylabel('Accuracy')
ax.set_title('Model Accuracy Comparison')
plt.ylim(0, 1) # Set the y-axis limits to 0-1 for accuracy
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

if __name__ == "__main__":
    preprocess_data_and_train_models()

```

## 9.2 level-2.ipynb

```
# Create Model Structure
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
class_count = len(list(train_gen.class_indices.keys())) # to define number of classes in dense
layer

# create pre-trained model (you can built on pretrained model such as : efficientnet, VGG ,
Resnet )
# we will use efficientnetb3 from EfficientNet family.
base_model = tf.keras.applications.efficientnet.EfficientNetB0(include_top= False, weights=
"imagenet", input_shape= img_shape, pooling= 'max')
base_model.trainable = False

model = Sequential([
    base_model,
    BatchNormalization(axis= -1, momentum= 0.99, epsilon= 0.001),
    Dense(256, kernel_regularizer= regularizers.l2(l= 0.016), activity_regularizer=
regularizers.l1(0.006),
        bias_regularizer= regularizers.l1(0.006), activation= 'relu'),
    Dropout(rate= 0.45, seed= 123),
    Dense(class_count, activation= 'softmax')
])

model.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy', metrics=
['accuracy'])

model.summary()

batch_size = 16 # set batch size for training
epochs = 5 # number of all epochs in training

history = model.fit(x= train_gen, epochs= epochs, verbose= 1, validation_data= valid_gen,
validation_steps= None, shuffle= False)
```

### 9.3 level-3.ipynb

```
from keras.models import Sequential
from keras.layers import Dense

# DL Model
dl_model = Sequential()
dl_model.add(Dense(32, input_dim=number_of_features, activation='relu'))
dl_model.add(Dense(16, activation='relu'))
dl_model.add(Dense(2, activation='softmax'))

dl_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

dl_model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10)

# ANN Model
ann_model = Sequential()
ann_model.add(Dense(32, input_dim=number_of_features, activation='relu'))
ann_model.add(Dense(16, activation='relu'))
ann_model.add(Dense(2, activation='softmax'))

# Compile the ANN model
ann_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the ANN model
ann_model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10)
import matplotlib.pyplot as plt

# Model names and accuracies
model_names = ["DL Model", "ANN Model"]
accuracies = [dl_accuracy, ann_accuracy]
# Create a bar chart to visualize the accuracy of each model
plt.figure(figsize=(8, 6))
plt.bar(model_names, accuracies, color=['blue', 'green'])
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.title('Model Accuracy Comparison')
plt.ylim(0, 1) # Set the y-axis limit from 0 to 1 for accuracy
plt.show()
```

## 10. OUTPUT SCREENS

### 10.1 GUI Home Page

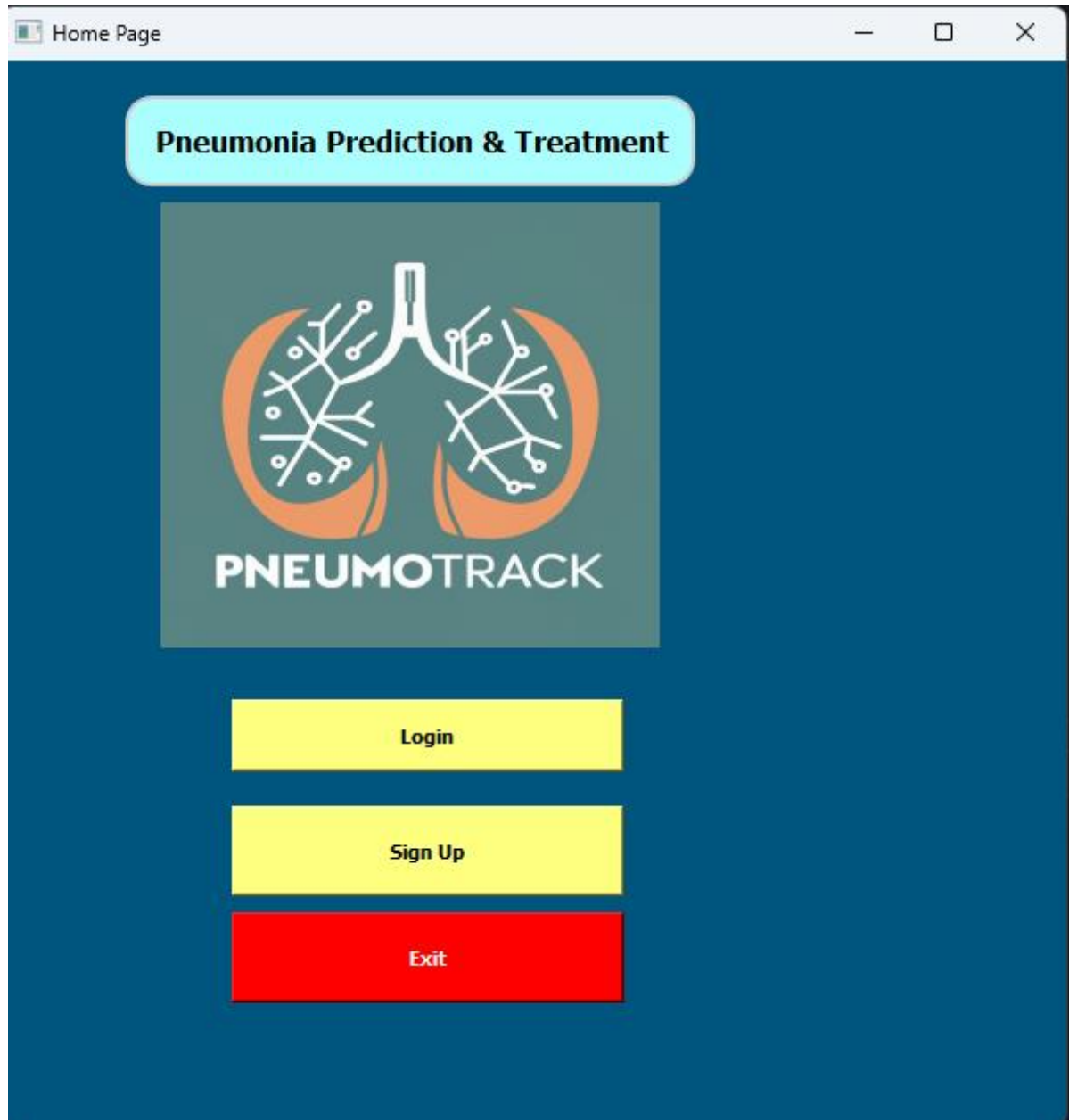
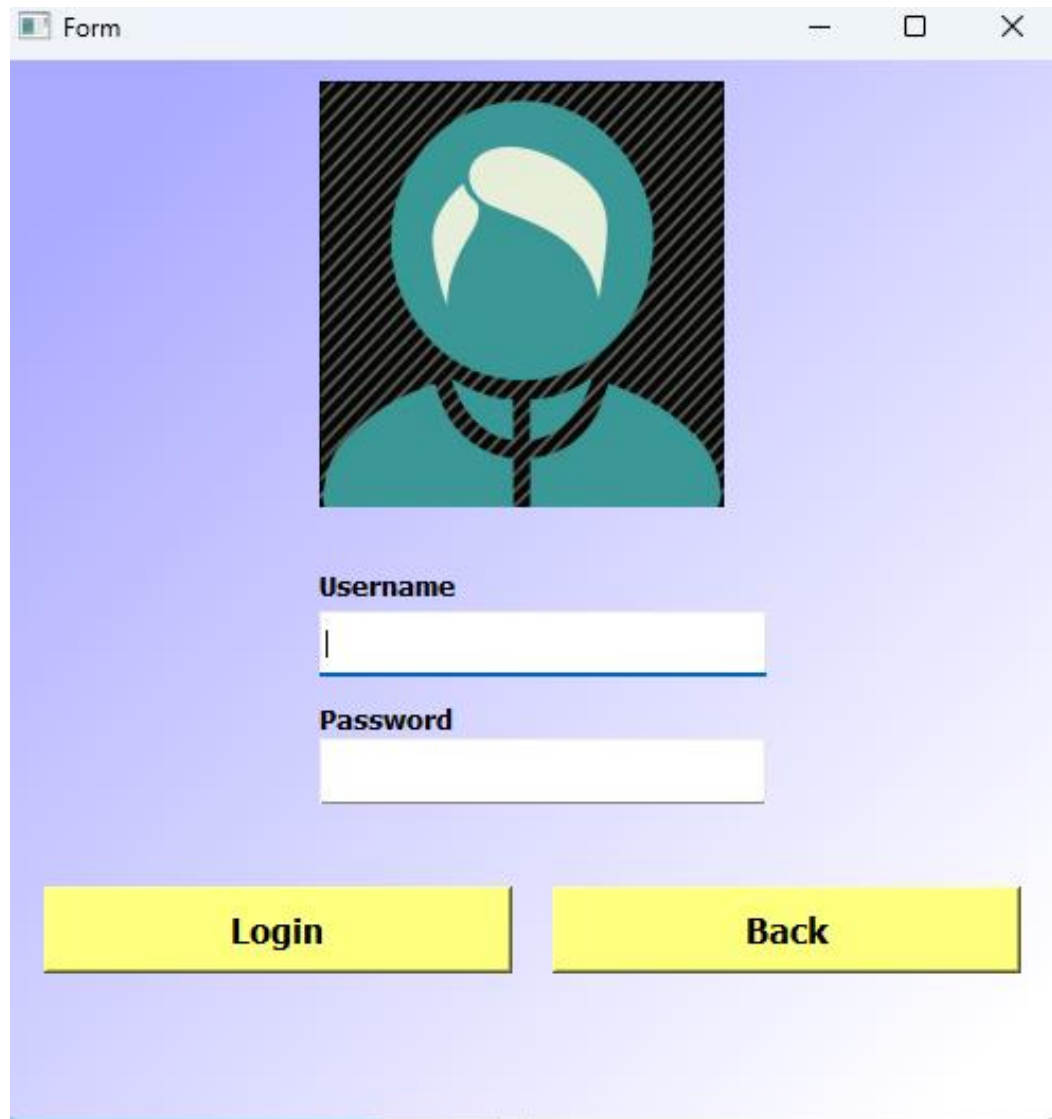


Fig. 10.1 GUI Home Page

## 10.2 Login Page



The image shows a web browser window with a title bar that says "Form". The page has a light blue background. At the top center is a square placeholder for a profile picture, showing a teal silhouette of a person's head and shoulders against a black background with diagonal lines. Below this, the word "Username" is written in bold black text, followed by a white text input field with a blue border. Underneath that, the word "Password" is written in bold black text, followed by another white text input field with a blue border. At the bottom of the form are two yellow buttons with black borders. The left button is labeled "Login" and the right button is labeled "Back".

Fig. 10.2 Login Page

## 10.3 SignUp Page

The image shows a web browser window with the title 'Form'. Inside the window is a registration form with a light blue background. At the top center of the form is a white rounded rectangle containing the text 'Registration Form'. Below this, there are five input fields arranged vertically. Each field has a label to its left: 'Name', 'Age', 'Gender', 'Username', and 'Password'. The input fields are light blue with a thin border. Below the input fields are two buttons. The first button is yellow with the text 'Register' in black. The second button is red with the text 'Cancel' in black. The browser window has standard minimize, maximize, and close buttons in the top right corner.

Fig. 10.3 Signup/ Registration page

## 10.4 Options Page

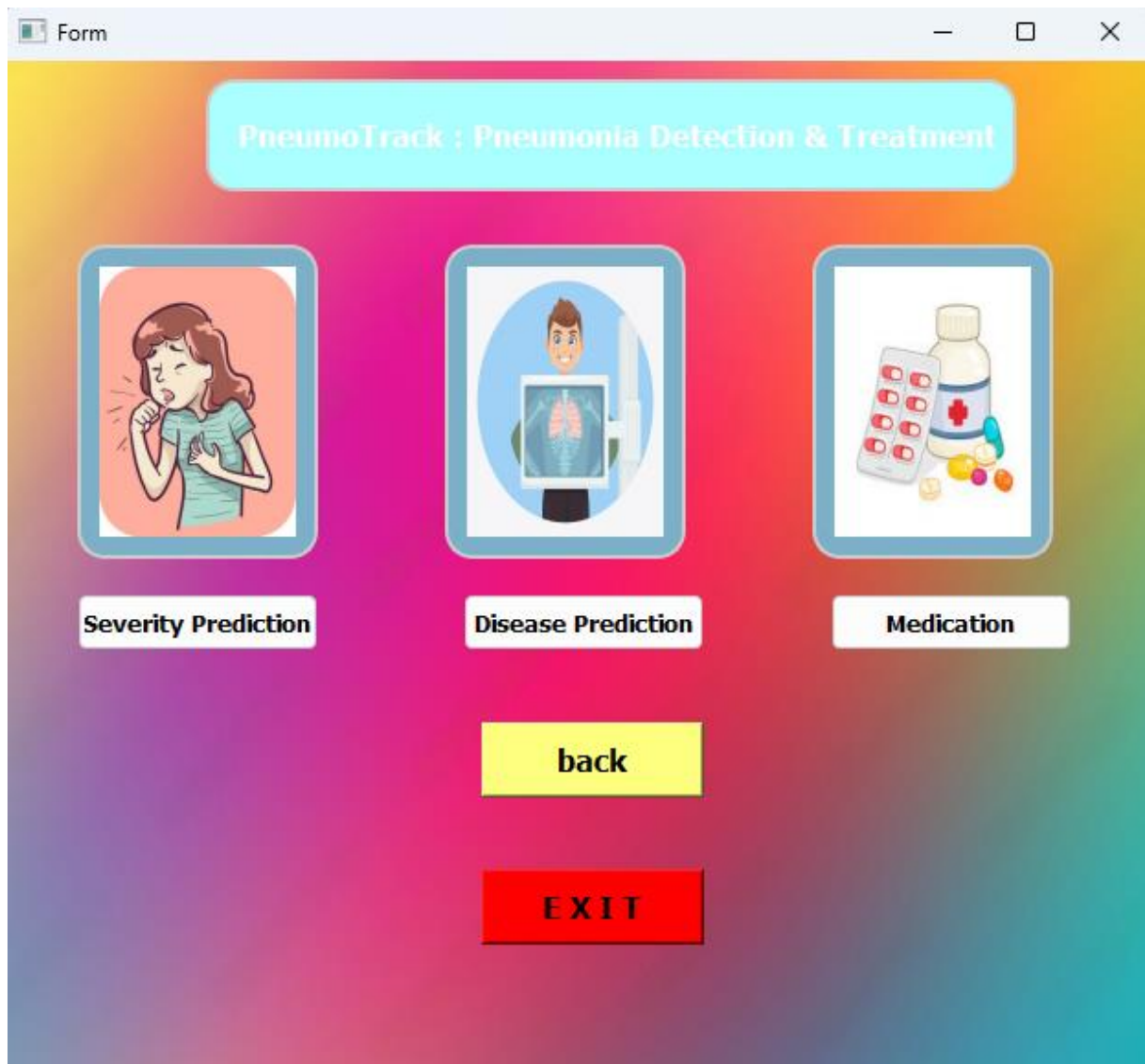



Fig. 10.4 Options Page

10.5 Severity Prediction Page

Form

Severity Prediction



Age

2

☐ Fatigue

☐ Headache

☐ Chills

☐ Sore Throat

☒ Fever

☐ Shortness of Breadth

☐ Cough

☐ Chest Pain

Severity Levels

0 - Mild

1 - Moderate

2 - Severe

Clear

Predict

2 - Severe

back

next

Fig. 10.5 Severity Prediction Page





## 10.6 Disease Prediction

Form

### Disease Prediction

Upload X-ray Image



Predict

Normal


back next

Fig. 10.6 Disease Prediction Page

## 10.7 Recommendations Page

Form

### Recommendations



**Severity** ☐ 0 ☐ 1 ☒ 2

**Diagnosis** ☒ Normal ☐ Pneumonia

**Recommendations**

**Clear**

Medications: Corticosteroids, Oxygen therapy  
Diet: Regular balanced diet  
Exercise: Light stretching

**back** **EXIT**

Fig. 10.7 Recommendations Page

## 11. CONCLUSION

In conclusion, this project represents a significant advancement in addressing the challenges posed by pneumonia, a life-threatening infectious disease that has a considerable impact on public health, particularly in regions like India. Our proposed advanced pneumonia detection and personalized treatment recommendation system leverages cutting-edge technologies to improve pneumonia management and empower individuals with valuable tools for self-care.

The project's objectives were meticulously designed to tackle various aspects of pneumonia diagnosis and treatment. We successfully developed a symptom-based prediction layer that assesses pneumonia severity based on user-reported symptoms, offering tailored initial treatment recommendations or guiding users towards further evaluation through X-ray image scans. Additionally, we implemented deep learning models to analyze chest X-ray images and classify the type of pneumonia, resulting in personalized treatment recommendations encompassing medications, diet plans, and exercise guidelines.

By implementing a user-friendly application, we've placed the power of pneumonia management directly into the hands of individuals, fostering early intervention and better disease management. However, it's imperative to emphasize that while our system serves as a valuable supportive tool, consulting healthcare professionals remains essential for accurate diagnosis and personalized treatment planning.

Through this project, we've made strides towards improving healthcare accessibility and the overall well-being of individuals impacted by pneumonia. As we move forward, further research, refinement, and collaboration with healthcare experts will be crucial to enhance the system's accuracy, expand its capabilities, and make a more significant positive impact on public health.

We believe that the fusion of advanced technology, data-driven insights, and healthcare expertise has the potential to revolutionize the way we approach pneumonia management, reducing its burden and improving outcomes for individuals across the country. This project represents a significant step towards realizing that vision, with the potential for broader applications in the field of healthcare.

## 12. FUTURE SCOPE

While this project represents a significant milestone in addressing pneumonia detection and management, there are several promising avenues for future development and enhancement:

1. **Integration of Advanced Imaging Techniques:** Consider integrating advanced medical imaging techniques such as CT scans or MRI for more precise and comprehensive pneumonia diagnosis. This could further improve the accuracy of classification and treatment recommendations.
2. **Real-time Monitoring:** Explore the possibility of real-time monitoring and tracking of pneumonia progression. This could involve continuous symptom assessment, wearable devices, and IoT integration to provide timely intervention and personalized care.
3. **Telemedicine Integration:** In the era of telemedicine, integrating your system with telehealth platforms could provide remote access to healthcare professionals for expert consultation and validation of diagnosis and treatment recommendations.
4. **Global Scalability:** Extend the reach of your system beyond regional boundaries, adapting it to different healthcare systems and languages. Localization and customization for diverse populations can make your project more accessible and impactful.
5. **Data Enrichment:** Continuously enhance your dataset with a focus on diverse patient demographics, comorbidities, and geographical variations. Robust data collection and augmentation can lead to more accurate predictions and recommendations.
6. **Machine Learning Interpretability:** Invest in research to improve the interpretability of machine learning models. Transparent models and clear decision-making processes are critical for gaining trust among healthcare professionals and users.

### 13. BIBLIOGRAPHY

1. A. K. Jaiswal, P. Tiwari, S. Kumar, D. Gupta, A. Khanna, and J. J. P. C. Rodrigues, "Identifying pneumonia in chest X-rays: a deep learning approach," *Measurement*, vol. 145, pp. 511–518, 2019.
2. A. Bhandary, G. A. Prabhu, V. Rajinikanth et al., "Deeplearning framework to detect lung abnormality—a study with chest X-ray and lung CT scan images," *Pattern Recognition Letters*, vol. 129, pp. 271–278, 2020.
3. G. Liang and L. Zheng, "A transfer learning method with deep residual network for pediatric pneumonia diagnosis," *Computer Methods and Programs in Biomedicine*, vol.187, Article ID 104964, 2019.
4. H. Behzadi-khormouji, H. Rostami, S. Salehi et al., "Deep learning, reusable and problem-based architectures for detection of consolidation on chest X-ray images," *Computer Methods and Programs in Biomedicine*, vol. 185, Article ID 105162, 2020.
5. I. Sirazitdinov, M. Kholiavchenko, T. Mustafaev, Y. Yixuan, R. Kuleev, and B. Ibragimov, "Deep neural network ensemble for pneumonia localization from a large-scale chest X-ray database," *Computers & Electrical Engineering*, vol. 78, pp. 388–399, 2019.
6. Jaiswal, A.K., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., Rodrigues, J.J. (2019). Identifying pneumonia in chest x-rays: a deep learning approach. *Measurement*, 145, 511–518.
7. Johnson, L. (2018). Deep learning applications in medical image analysis (Doctoral dissertation). Stanford University.
8. M. D. Alshehri and F. K. Hussain, "A fuzzy security protocol for trust management in the internet of things (fuzzy-IoT)," *Computing*, vol. 101, no. 7, pp. 791–818, 2019.
9. M. D. Alshehri, F. Hussain, M. Elkhodr, and B. S. Alsinglawi, "A distributed trust management model for the internet of things (DTM-IoT)," *Recent Trends and Advances in Wireless and IoT-Enabled Networks*, Springer, Berlin, Germany, 2019.
10. M. Toğaçar, B. Ergen, Z. Cömert, and F. Ozyurt, "A deep feature learning model for pneumonia detection applying a combination of MRMR feature selection and machine learning models," *IRBM*, vol. 41, 2019.

11. Orbann, C., Sattenspiel, L., Miller, E., Dimka, J. (2017). Defining epidemics in computer simulation models: how do definitions influence conclusions? *Epidemics*, 19\*, 24–32.
12. TensorFlow (2021). TensorFlow: An open-source machine learning framework (Version 2.7.0) [Software]. <https://www.tensorflow.org/>
13. UNICEF. (2019). Child Health - Pneumonia. <https://data.unicef.org/topic/child-health/pneumonia/> accessed on 15 July 2019.
14. World Health Organization. Pneumonia. <https://www.who.int/news-room/fact-sheets/detail/pneumonia>.
15. Y. Ge, Q. Wang, L. Wang et al., “Predicting post-stroke pneumonia using deep neural network approaches,” *International Journal of Medical Informatics*, vol. 132, Article ID 103986, 2019.